University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Paweł Betliński

Bayesian Network Approximation from Local Structures

PhD dissertation

Supervisor
dr hab. Dominik Ślęzak
Institute of Informatics, University of Warsaw

June 2016

Author's declaration

Aware of legal responsibility I hereby declare that I have written this dissertation myself and all the contents of the dissertation have been obtained by legal means.

date Author's signature

Supervisor's declaration

The dissertation is ready to be reviewed.

date Supervisor's signature

Bayesian Network Approximation from Local Structures

Abstract

This work is focused on the problem of Bayesian network structure learning. There are two main areas in this field which are here discussed.

The first area is a theoretical one. We consider some aspects of the Bayesian network structure learning hardness. In particular we prove that the problem of finding a Bayesian network structure with a minimal number of edges encoding the joint probability distribution of a given dataset is NP-hard. This result can be considered as a significantly different than the standard one view on the NP-hardness of the Bayesian network structure learning. The most notable so far results in this area are focused mainly on the specific characterization of the problem, where the aim is to find a Bayesian network structure maximizing some given probabilistic criterion. These criteria arise from quite advanced considerations in the area of statistics, and in particular their interpretation might be not intuitive—especially for the people not familiar with the Bayesian networks domain. In contrary the proposed here criterion, for which the NP-hardness is proved, does not require any advanced knowledge and it can be easily understandable.

The second area is related to concrete algorithms. We focus on one of the most interesting branch in history of Bayesian network structure learning methods, leading to a very significant solutions. Namely we consider the branch of local Bayesian network structure learning methods, where the main aim is to gather first of all some information describing local properties of constructed networks, and then use this information appropriately in order to construct the whole network structure. The algorithm which is the root of this branch is focused on the important local characterization of Bayesian networks—so called Markov blankets. The Markov blanket of a given attribute consists of such other attributes which in the probabilistic sense correspond to the maximal in strength and minimal in size set of its causes. The aforementioned first algorithm in the considered here branch is based on one important observation. Subject to appropriate assumptions it is possible to determine the optimal Bayesian network structure by examining relations between attributes only within the Markov blankets. In the case of datasets derived from appropriately sparse distributions, where Markov blanket of each attribute has a limited by some common constant size, such procedure leads to a well time scalable Bayesian network structure learning approach. The Bayesian network local learning branch has mainly evolved in direction of reducing the gathered local information into even smaller and more reliably learned patterns. This reduction has raised from the parallel progress in the Markov blankets approximation field.

The main result of this dissertation is the proposal of Bayesian network structure learning procedure which can be placed into the branch of local learning methods and which leads to the fork in its root in fact. The fundamental idea is to appropriately aggregate learned over the Markov blankets local knowledge not in the form of derived dependencies within these blankets—as it happens in the root method, but in the form of local Bayesian networks. The user can thanks to this have much influence on the character of this local knowledge—by choosing appropriate to his needs Bayesian network structure learning method used in order to learn the local structures. The merging approach of local structures into a global one is justified theoretically and evaluated empirically, showing its ability to enhance even very advanced Bayesian network structure learning algorithms, when applying them locally in the proposed scheme.

Keywords

Bayesian network, Markov blanket, NP-hardness, conditional independence, d-separation, local learning

ACM Classification (2012):

Computing methodologies → Machine learning → Machine learning approaches → Learning in probabilistic graphical models → Bayesian network models

Aproksymacja Sieci Bayesowskiej ze Struktur Lokalnych

Streszczenie

Praca ta skupia się na problemie uczenia struktury sieci bayesowskiej. Są dwa główne pola w tym temacie, które są tutaj omówione.

Pierwsze pole ma charakter teoretyczny. Rozpatrujemy pewne aspekty trudności uczenia struktury sieci bayesowskiej. W szczególności pokozujemy, że problem wyznaczenia struktury sieci bayesowskiej o minimalnej liczbie krawędzi kodującej w sobie łączny rozkład prawdopodobieństwa atrybutów danej tabeli danych jest NP-trudny. Rezultat ten może być postrzegany jako istotnie inne od standardowego spojrzenie na NP-trudność uczenia struktury sieci bayesowskiej. Najbardziej znaczące jak dotąd rezultaty w tym zakresie skupiają się głównie na specyficznej charakterystyce problemu, gdzie celem jest wyznaczenie struktury sieci bayesowskiej maksymalizującej pewne zadane probabilistyczne kryterium. Te kryteria wywodzą się z dość zaawansowanych rozważań w zakresie statystyki i w szczególności mogą nie być intuicyjne—szczególnie dla ludzi niezaznajomionych z dziedziną sieci bayesowskich. W przeciwieństwie do tego zaproponowane tutaj kryterium, dla którego została wykazana NP-trudność, nie wymaga żadnej zaawansowanej wiedzy i może być łatwo zrozumiane.

Drugie pole wiąże się z konkretnymi algorytmami. Skupiamy się na jednej z najbardziej interesujących gałęzi w historii metod uczenia struktur sieci bayesowskich, prowadzącej do bardzo znaczących rozwiązań. Konkretnie rozpatrujemy gałąź metod lokalnego uczenia struktur sieci bayesowskich, gdzie głównym celem jest zebranie w pierwszej kolejności pewnych informacji opisujących lokalne własności konstruowanych sieci, a następnie użycie tych informacji w odpowiedni sposób celem konstrukcji pełnej struktury sieci. Algorytm będący korzeniem tej gałęzi skupia się na ważnej lokalnej charakteryzacji sieci bayesowskich—tak zwanych kocach Markowa. Koc Markowa dla zadanego atrybutu składa się z tych pozostałych atrybutów, które w sensie probabilistycznym odpowiadają maksymalnymu w sile i minimalnemu w rozmiarze zbiorowi jego przyczyn. Wspomniany pierwszy algorytm w rozpatrywanej tu gałęzi opiera się na jednej istotnej obserwacji. Przy odpowiednich założeniach możliwe jest wyznaczenie optymalnej struktury sieci bayesowskiej poprzez badanie relacji między atrybutami jedynie w obrębie koców Markowa. W przypadku zbiorów danych wywodzących się z odpowiednio rzadkiego rozkładu, gdzie koc Markowa każdego atrybutu ma ograniczony przez pewną wspólną stałą rozmiar, taka procedura prowadzi do dobrze skalowalnego czasowo podejścia uczenia struktury sieci bayesowskiej. Gałąź lokalnego uczenia sieci bayesowskich rozwinęła się głównie w kierunku redukcji zbieranych lokalnych informacji do jeszcze mniejszych i bardziej niezawodnie wyuczanych wzorców. Redukcja ta wyrosła na bazie równoległego rozwoju w dziedzinie aproksymacji koców Markowa.

Głównym rezultatem tej rozprawy jest zaproponowanie procedury uczenia struktury sieci bayesowskiej, która może być umiejscowiona w gałęzi metod lokalnego uczenia i która faktycznie wyznacza rozgałęzienie w jego korzeniu. Fundamentalny pomysł polega tu na tym, żeby odpowiednio agregować wyuczoną w obrębie koców Markowa lokalną wiedzę nie w formie wyprowadzonych zależności w obrębie tych koców—tak jak to się dzieje w przypadku metody - korzenia, ale w formie lokalnych sieci bayesowskich. Użytkownik może mieć dzięki temu duży wpływ na charakter tej lokalnej wiedzy—poprzez wybór odpowiedniej dla jego potrzeb metody uczenia struktury sieci bayesowskiej użytej w celu wyznaczenia lokalnych struktur. Procedura scalenia lokalnych modeli celem utworzenia globalnego jest uzasadniona teoretycznie oraz zbadana eksperymentalnie, pokazując jej zdolność do poprawienia nawet bardzo zaawansowanych algorytmów uczenia struktury sieci bayesowskiej, gdy zastosuje się je lokalnie w ramach zaproponowanego schematu.

Słowa kluczowe

sieć bayesowska, koc Markowa, NP-trudność, niezależność warunkowa, d-separacja, lokalne uczenie

Klasyfikacja ACM (2012)

Computing methodologies → Machine learning → Machine learning approaches → Learning in probabilistic graphical models → Bayesian network models

Contents

Chapter 1

Introduction

A Bayesian network (Pearl, 1985, 1988) is a directed acyclic graph encoding the joint probability distribution of the random variables represented by the vertices of this graph. In a common scenario these variables have only finite sets of possible values. This distribution information is held by means of the parameters of the graph describing the probability distribution of each of the vertices given its parents in the graph.

The following two observations about applications of Bayesian networks are often considered.

First of all, one of the fundamental features of Bayesian networks is their ability to visualize in a way easily understandable by humans dependencies between the represented variables. Roughly speaking the parents of any variable in a Bayesian network can be seen as the direct causes of this variable. The induced from a data table Bayesian network represents in an approximate way the joint distribution of the attributes in the table. From the point of view of simplicity in perception of induced networks by a human as well as the minimum description length principle, one of the main desired properties of these networks is their sparsity, partially understood as the sparsity of their structures, that is the sparsity of corresponding to them graphs.

A sparser network representing approximately a desired data distribution tends to have the set of direct causes (parents) of each attribute closer to the minimal possible correct choice. Hence, a sparser network represents the dependencies occurring in a considered domain in a compact way, including only these in fact important attributes relationships.

Secondly, a Bayesian network, especially a sparse network, can represent joint probability distributions in a significantly compressed way. Direct representation of a joint probability distribution requires storing an exponential with regard to number of variables amount of parameters. Meanwhile a sparse network, for example with the bounded number of parents of each vertex, requires storing only a linear amount of parameters. This is a significant difference in the case of distributions which can be encoded by such structure.

Learning a high quality, as sparse as possible, Bayesian network representing the dependencies of a given dataset is itself a challenging task, especially if the scalability of a learning method with respect to the increasing number of attributes is required. Very interesting is in particular a Bayesian network structure learning process, that is determining just a DAG representing a desired network.

One of the main classes of Bayesian network structure learning methods is the class of score-based methods, where the theoretical aim, typically unreachable in practical applications, is to find a network structure maximizing some scoring criterion. There has been shown the NP-hardness of such task in several important aspects, from which the most notable are

Chickering et al. results (Chickering, 1996; Chickering et al., 2004).

An alternative for the score-based methods, the second main class of Bayesian network structure learning algorithms, is the class of constraint-based methods, where a whole Bayesian network structure is deduced with an assist of statistical tests for conditional independence. The fundamental idea here is to obtain from these tests operating on appropriately chosen subsets of attributes the knowledge about so called d-separations appearing in a perfect map—a structure of an optimal in some sense for a considered data distribution Bayesian network. An induced set of d-separations can, at least in theoretical conditions, where we assume that the obtained knowledge about them is in 100 percent proper, clearly determine a desired network.

1.1. Main Contributions

This dissertation is focused on two aspects.

The first one is related to the hardness of Bayesian network structure learning. We establish here a new and significantly different than the classical one view on the NP-hardness of this learning. The proposed view operates on very simple concepts, which results in an intuitive and easy to understand statement. It tries to take advantage of both the aforementioned most recognizable NP-hardness results. Namely, it is aimed at the purely practical scenario of learning the network structure on the basis of some available finite sample similarly as the Chickering (1996) approach, but in the same time it sticks to the much less complicated and easy to interpret optimization criterion—similarly as the Chickering et al. (2004) approach. The whole presented here idea has raised from the theoretical considerations about the related to Markov blankets approximate versions of decision reducts (Ślęzak, 2002) known from the theory of rough sets (Pawlak and Skowron, 2007).

The second aspect is related to the branch of local Bayesian network structure learning algorithms. The main result of this dissertation is a proposed here new solution belonging to this branch, which in particular is a significant generalization of the root method—so called GS (Grow-Shrink) algorithm (Margaritis and Thrun, 2000).

The Grow-Shrink is the name of in fact two algorithms: the Markov blanket detection algorithm and strictly related to it Bayesian network structure learning method. The structure learning method performs the induction of the whole network from the knowledge about previously learned with the assist of the first algorithm Markov blankets and some additionally acquired in the constraint-based manner local relationships within these blankets.

The algorithm proposed in the dissertation is a generalization of this structure learning method in the sense that it changes the character of locally acquired knowledge—converting the strict form of its gathering into the one arbitrary chosen by the user. The local information within the blankets is here in the form of local, operating on subsets of attributes corresponding to these blankets, Bayesian networks. Any already existing Bayesian network structure learning method can be used in order to gather this local knowledge.

The proposed LBNA (Local Bayesian Networks Aggregation) mechanism is a procedure of merging these locally induced structures into the whole Bayesian network operating on all attributes. It is justified theoretically and evaluated empirically. This mechanism opens completely new possibilities of a kind of boosting of the existing structure learning methods—when we apply them locally in the proposed scheme. In the experiments chapter we have in particular exposed strong evidences showing how one of the most significant Bayesian network structure learning methods, also belonging to the branch of local learning algorithms—the MMHC (Max-Min Hill-Climbing) approach (Tsamardinos et al., 2006), can be improved with

an assist of the LBNA methodology.

1.2. Work Organization

The work is organized as follows.

Chapter 2 presents formally some basic concepts and several fundamental theoretical results related to the Bayesian networks domain. It in particular introduces most of the notation used in the remaining part of this work. It also introduces and explains what is the idea and motivation standing behind one of the two main classes of Bayesian network structure learning methods, the class of constraint-based learning algorithms. As a consequence it investigates the notion of conditional independence and the problem of its practical detection with an assist of statistical tests.

Chapter 3 presents some of the most notable results related to the hardness of Bayesian network inference and learning—namely the Cooper (1990), Chickering (1996) and Chickering et al. (2004) results. These results are considered nowadays as a most significant view in this topic. It also introduces the second main class of Bayesian network learning methods, the class of score-based learning algorithms. The reason is that the main theoretical results related to the hardness of Bayesian network learning are placed exactly in the score-based paradigm. The chapter is finished with an alternative and proposed in this work view on the Bayesian network learning hardness.

Chapter 4 gives a detailed overview of some of the most fundamental constraint-based learning algorithms. We in particular describe in details two first constraint-based learning approaches: SGS (Spirtes et al., 1990) and PC (Spirtes and Glymour, 1991), and one aforementioned very significant improvement of them—the GS (Grow-Shrink) method (Margaritis and Thrun, 2000). GS exploits in contrary to any previous network structure learning method the notion of Markov blankets—which was a very successful choice.

GS became the inspiration and the root method of the whole branch of local learning algorithms. This branch is the main topic of Chapter 5. We describe there first some intermediate enhancements and discoveries in this topic, like the IAMB (Incremental Association Markov Blanket) algorithm (Tsamardinos et al., 2003a) and MMPC (Max-Min Parents and Children) algorithm (Tsamardinos et al., 2003b). We finish with the great result of research within this field—the MMHC (Max-Min Hill-Climbing) method (Tsamardinos et al., 2006).

Chapter 6 introduces the algorithm LBNA (Local Bayesian Networks Aggregation)—which is the main result of this work. Firstly some motivation and innovativeness aspects of the proposed approach with respect to the branch of local Bayesian network learning methods are discussed. Then the main theoretical result on which the LBNA approach is based is presented and explained—together with some possible enhancements and modifications. A detailed explanation of the proposed on the basis of this result Bayesian network induction methodology LBNA is given. Also a detailed analysis of the approach with respect to several aspects is presented.

Chapter 7 describes in details all the experiments performed with the proposed algorithm. Quite large number of detailed statistics have been placed and appropriately grouped in Appendix A in order to enhance readability of the whole work. Chapter 7 contains some summarizing figures instead, on the basis of which in particular strong conclusions are derived.

Chapter 8 summarizes the whole dissertation and sketches some possible directions of the future work.

Appendix B contains the listing of most of the notation used in this work, with the references to the appropriate parts of this thesis.

Chapter 2

Basic Concepts Related to Bayesian Networks

In this chapter most of the terminology used in the remaining part of the dissertation is introduced, as well as most of the further referred theoretical results. First we describe some fundamental elements of the environment in which all this dissertation is placed, like datasets, random variables, their distributions and graphs. We state some assumptions regarding these elements. These assumptions hold in all further chapters. We explain also some basic notation rules used in all the work. Then we recall the notion of conditional independence, some practical aspects related to testing such independence, and a very significant in the domain of Bayesian networks term expressing some specific conditional independence relation—so called Markov blanket. Then we formally introduce the concept of Bayesian network, together with several fundamental theoretical results related to it. We finish with the introduction of the constraint-based Bayesian network learning class of algorithms. This is a crucial for us class, as the branch of local Bayesian network structure learning methods, including the proposed here approach, clearly belongs to it.

2.1. Dataset Interpretations

We describe in this section several main elements of the environment in which this work is placed. All of them are describing from different points of view the same concept: dataset. First we formalize this notion as the information system , then we discuss the probabilistic representation of its attributes in the form of distribution, and we introduce the statistical view on the dataset as the sample from the given distribution. Finally we sketch first intuitions about the graphical representation of the given dataset, or rather of the distribution the sample of which is available for us in the form of the dataset.

2.1.1. Information System

A Bayesian network is one of the famous probabilistic tools used in the data exploration domain. Very often in the case of this domain on the entry we have some dataset. We can express it by the notion of an information system (Pawlak, 1983).

Definition 2.1.1.1 (information system) *An information system is a tuple* $(\mathcal{U}, \mathcal{V})$*,where* $\mathcal{U}$ *is a finite set of so called objects and* $\mathcal{V}$ *is a finite set of so called attributes.*

A dataset is often represented as a table, where the columns correspond to attributes and rows correspond to objects. The interpretation of objects is that they represent some specific cases, elements of somehow defined group, for example patients of some hospital. Attributes give a kind of description of each such object. Each attribute corresponds to some feature of the object, for example the age or weight of the patient.

We will further interchangeably use notions of the information system and dataset, attribute and column, as well as object and row.

$\mathcal{D}$	H	B	L	F	C
U_1	$h1$	$b2$	$l2$	$f1$	$c1$
U_2	$h1$	$b2$	$l1$	$f1$	$c2$
U_3	$h2$	$b1$	$l2$	$f2$	$c1$
U_4	$h2$	$b2$	$l2$	$f1$	$c2$
U_5	$h1$	$b1$	$l1$	$f1$	$c1$
U_6	$h1$	$b2$	$l2$	$f1$	$c1$
U_7	$h2$	$b2$	$l2$	$f1$	$c1$
U_8	$h1$	$b1$	$l1$	$f2$	$c1$
U_9	$h2$	$b1$	$l1$	$f1$	$c2$
U_{10}	$h1$	$b1$	$l2$	$f2$	$c2$

Attribute	Value	Description
H	$h1$	there is a history of smoking
H	$h2$	there is no history of smoking
B	$b1$	bronchitis is present
B	$b2$	bronchitis is absent
L	$l1$	lung cancer is present
L	$l2$	lung cancer is absent
F	$f1$	fatigue is present
F	$f2$	fatigue is absent
C	$c1$	chest X-ray is positive
C	$c2$	chest X-ray is negative

Figure 2.1.1.1: The table on the left is an exemplary dataset $\mathcal{D}$ containing some information about the patients diagnosed with the chest X-ray. More formally, it is an information system $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ where $\mathcal{U} = \{U_1, ..., U_{10}\}$ is the set of diagnosed patients and $\mathcal{V} = \{H, B, L, F, C\}$ is the set of attributes describing them. The table on the right contains the information about the possible values of the attributes and the interpretation of each such value.

Each attribute have some possible values. In all this work we limit ourselves to the case where all attributes are categorical, that is each attribute can have only some finite number of possible values—but at least two (we do not take into account the trivial columns, which do not interact anyhow with the remaining attributes). For example in the case of weight of the patient instead of expressing it directly in the form of some decimal number we would here stick only to the case where such weight is expressed only in terms of some finite number of possible categories, for example: "thin", "normal" and "fat". An example dataset reflecting some information about the patients diagnosed with the chest X-ray is presented in Figure 2.1.1.1.

Let us introduce some notation which will enable us in further chapters to easily describe some specific parts of the given dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$. The value in $\mathcal{D}$ corresponding to the object $U \in \mathcal{U}$ and attribute $V \in \mathcal{V}$ we will denote as $\mathcal{D}[U, V]$. For example, if we consider the dataset $\mathcal{D}$ given in Figure 2.1.1.1, then $\mathcal{D}[U_5, L] = l1$.

For the given object $U \in \mathcal{U}$ and subset of attributes $\mathbb{V} \subseteq \mathcal{V}$ by $\mathcal{D}[U, \mathbb{V}]$ we will understand the values assignment of the attributes subset $\mathbb{V}$ for the object U. In the case of the dataset $\mathcal{D}$ from Figure 2.1.1.1 we would for example have that $\mathcal{D}[U_4, \{H, B, C\}] = \{h2, b2, c2\}$.

For the given two subsets $\mathbb{U} \subseteq \mathcal{U}$ and $\mathbb{V} \subseteq \mathcal{V}$ by $\mathcal{D}[\mathbb{U}, \mathbb{V}]$ we will understand a dataset truncated to the given in the arguments set of objects and attributes. According to our example dataset $\mathcal{D}$ in Figure 2.1.1.1, the dataset $\mathcal{D}' = \mathcal{D}[\{U_3, U_6, U_9\}, \{B, F\}]$ would look as follows:

$\mathcal{D}'$	B	F
U_3	$b1$	$f2$
U_6	$b2$	$f1$
U_9	$b1$	$f1$

2.1.2. Distribution

In the probability theory and in particular in the statistics attributes of a given dataset are interpreted as the random variables. In our case each attribute will correspond to a discrete random variable, which has only some finite number of possible values, but at least two.

Let us point it out very clearly here, that everywhere further in this dissertation it is silently assumed, that we deal with a finite sets of discrete random variables or corresponding attributes—which have only some finite number of possible values, but at least two.

We will interchangeably use the same notation for the variables and corresponding to them attributes. Referring to our exemplary dataset $\mathcal{D}$ presented in Figure 2.1.1.1, depending on the context, the sign B will sometimes mean the attribute, while in some other case the random variable corresponding to this attribute. The context of use should be always clear, or appropriately stated in the case of possible ambiguity.

Here and everywhere further we denote the set of all considered random variables, or the set of all corresponding to them attributes in a given dataset, with the sign $\mathcal{V}$—and again the appropriate meaning of $\mathcal{V}$ will always depend on the context of use.

The most important for us characterization of a given set of random variables $\mathcal{V} = \{X_1, ..., X_n\}$ will be its joint probability distribution, which we denote always as $\mathcal{P}$.

$\mathcal{P}$ is the knowledge about the probability of each event corresponding to the set of random variables $\mathcal{V}$. For example, if $\mathcal{V}$ was the set of variables corresponding to the attributes of our dataset $\mathcal{D}$ from Figure 2.1.1.1, then the joint probability distribution of these variables $\mathcal{P}$ would express in particular:

- the probability of each possible values assignment to all these variables, for example: $\mathcal{P}(H = h2, B = b1, L = l1, F = f2, C = c1)$. The fundamental property of any joint probability distribution $\mathcal{P}$ is that the sum of probabilities of all possible values configurations of all variables is equal to 1. This information is all we need to have in order to fully characterize the given distribution. The following points can be easily retrieved from this knowledge,

- the probability of each possible values assignment to any subset of these variables, for example: $\mathcal{P}(B = b2, F = f1)$,

- any conditional probability regarding arbitrary chosen subsets of these variables, for example: $\mathcal{P}(B = b2, F = f1 \mid H = h2, C = c1)$.

In each of the above expressions we use the same sign $\mathcal{P}$—and this will be the convention used everywhere further.

We will occasionally write some probability equations in a more generative form, without specifying explicit values of occurring variables. Such equations should be understood as follows—the specified in the equation property holds for every explicit values assignment of the occurring variables. For example, in order to express that two subsets of random variables $\mathbb{X} = \{X_1, ..., X_m\}$ and $\mathbb{Y} = \{Y_1, ..., Y_n\}$ of the set of all variables $\mathcal{V}$ (we denote always subsets of $\mathcal{V}$ with the capital blackboard letters) having joint probability distribution $\mathcal{P}$ are independent we would write:

$$\mathcal{P}(X_1, ..., X_m, Y_1, ..., Y_n) = \mathcal{P}(X_1, ..., X_m)\mathcal{P}(Y_1, ..., Y_n).$$

A given dataset represents itself some joint probability distribution of its attributes, by means of frequencies occurring in the table. We will denote this distribution with the symbol $\mathcal{P}$ with the index corresponding to the dataset, for example $\mathcal{P}_{\mathcal{D}}$ for the case of dataset $\mathcal{D}$. In

such distribution the probability of any values assignment to all variables-attributes is equal to the number of rows in the table where such assignment is present divided by the number of all rows in the table. In the analogical fashion all other (conditional) probabilities can be derived. For example, if we look at our dataset $\mathcal{D}$ from Figure 2.1.1.1, then we can calculate on its bases the following properties of its distribution $\mathcal{P}_{\mathcal{D}}$:

- $\mathcal{P}_{\mathcal{D}}(H = h1, B = b2) = 0.3$, as there are three out of ten rows in this table where $H = h1$ and $B = b2$,
- $\mathcal{P}_{\mathcal{D}}(H = h1 \mid B = b2, L = l2) = 0.5$, as there are two rows in the table where $H = h1$, $B = b2$ and $L = l2$ out of four rows where $B = b2$ and $L = l2$.

2.1.3. Sample

A typical in the Bayesian networks domain assumption connecting a given dataset $\mathcal{D}$ and a corresponding to its attributes set of variables $\mathcal{V}$ is that a dataset is a sample generated from $\mathcal{P}$—the joint probability distribution of $\mathcal{V}$.

Definition 2.1.3.1 (sample, generative distribution) *The sample of size n generated from the joint probability distribution $\mathcal{P}$ of a given set of variables $\mathcal{V}$ we understand as such dataset $\mathcal{D}$ which has set of attributes $\mathcal{V}$ and n rows, where each row consists of randomly with regard to the distribution $\mathcal{P}$ chosen values assignment of variables in $\mathcal{V}$. The random choice for each next row is independent from the random choices of remaining rows.*

In the case when a dataset $\mathcal{D}$ is the sample generated from some joint probability distribution $\mathcal{P}$ we say that $\mathcal{P}$ is a generative distribution for $\mathcal{D}$.

The existence of some generative distribution for the given dataset is a theoretical condition, which might be often not true in practice. The most common situation when this assumption is not valid occurs in the situation where the generating process of the dataset is not stationary, when it is for example somehow changing in time.

However, the basic assumption in the Bayesian networks domain is that such generative distribution exists.

2.1.4. Graph

Section 2.3 explains formally what a Bayesian network is. Now let us only mention that it is in fact a compressed representation of the above mentioned generative distribution. In practice, where such network is obtained in the process of learning based on the available sample—it is a compressed approximate representation of this theoretical distribution—simply because as the input only some finite dataset usually does not reflecting perfectly a generative distribution is available. In other words, if $\mathcal{D}$ is a sample generated from some joint probability distribution $\mathcal{P}$, then it should be expected that the distributions $\mathcal{P}$ and $\mathcal{P}_{\mathcal{D}}$ will be to some extend different.

One of the components of every Bayesian network is its so called structure—a graph representing dependencies occurring within a given set of variables $\mathcal{V} = \{X_1, ..., X_n\}$. It is a directed, acyclic graph (DAG), consisting of nodes exactly corresponding to the variables in the set $\mathcal{V}$.

The set of nodes occurring in this graph we will denote with the same symbol $\mathcal{V}$, and the particular nodes as X_1, ..., X_n. This means that all these symbols can have one of the three meanings, depending on the context: random variables, attributes of the corresponding to

these variables sample, or nodes of the graph representing the Bayesian network visualizing the distribution of these variables.

For example, for the case of the dataset presented in Figure 2.1.1.1 one can consider the Bayesian network structure given in Figure 2.1.4.1. Let us finish the section with this picture. Its formal interpretation will become understandable during Section 2.3.

An informal interpretation is that in a Bayesian network structure parents of any node correspond to its direct causes. What we mean by a direct cause will be also the further topic. But some intuitive understanding is easy to catch. For example, if we look at the node C which corresponds to the attribute expressing the result of the chest X-ray diagnoses—it seems to be natural that the direct influence on it has the attribute L, expressing whether a given patient has the lung cancer. And indeed the node L is the only one parent of C.

As a second example let us consider the node F, which corresponds to the attribute expressing whether for a given patient a fatigue is present. Its parents in the graph are B and L. And in fact these two attributes, representing the currently present for the case of a patient illnesses, seem to be its direct causes.

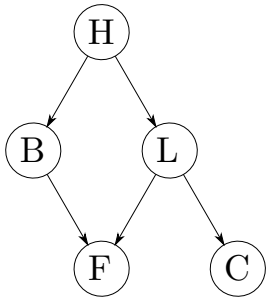


Figure 2.1.4.1: An exemplary Bayesian network structure corresponding to the dataset presented in Figure 2.1.1.1.

2.2. Conditional Independence—Detection in Theory and Practice

This section is about the most important from the point of view of further chapters notion, called conditional independence. The main aspect on which we are here focused is the practical detection of conditional independence, by means of statistical tests. These tests are the main bricks of every constraint-based Bayesian network structure learning algorithm, while exactly this class of algorithms is the main point of interest in this dissertation. The section is finished with here only brief introduction of the Markov blanket concept—which corresponds to the specific conditional independence relation. We return to this notion later and reveal its deep connections with the Bayesian networks domain. Markov blankets are main bricks of several constraint-based Bayesian network structure learning algorithms from the especially interesting for us branch of local learning methods, including the proposed in this work approach.

2.2.1. Conditional Independence Definition

A very extensively appearing in the domain of Bayesian networks notion is the conditional independence. It expresses whether some two events does not influence on each other when

we have the knowledge about some third (conditional) event.

Our reference point here and everywhere further will be the finite set $\mathcal{V}$ of some random variables, where the set of possible values of each variable is finite. Such set of possible values of a given random variable $X \in \mathcal{V}$ we will denote as Dom_X. The set of possible values assignments of a given set of random variables $\mathbb{X} \subseteq \mathcal{V}$ we will denote as $Dom_{\mathbb{X}}$.

The conditional independence occurring within the set $\mathcal{V}$ we will understand as follows.

Definition 2.2.1.1 (conditional independence, $Ind_{\text{distribution}}$ notation) *Let $\mathcal{V}$ be a set of random variables, and let $\mathcal{P}$ be its joint probability distribution. Let us consider three mutually exclusive nonempty subsets of variables $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subset \mathcal{V}$. We say that $\mathbb{X}$ and $\mathbb{Y}$ are conditionally independent given $\mathbb{Z}$ according to the distribution $\mathcal{P}$, if for all $\mathtt{x} \in Dom_{\mathbb{X}}$, $\mathtt{y} \in Dom_{\mathbb{Y}}$ and $\mathtt{z} \in Dom_{\mathbb{Z}}$ where $\mathcal{P}(\mathbb{Z} = \mathtt{z}) > 0$ it holds that:*

$$\mathcal{P}(\mathbb{X} = \mathtt{x}, \mathbb{Y} = \mathtt{y} \mid \mathbb{Z} = \mathtt{z}) = \mathcal{P}(\mathbb{X} = \mathtt{x} \mid \mathbb{Z} = \mathtt{z})\mathcal{P}(\mathbb{Y} = \mathtt{y} \mid \mathbb{Z} = \mathtt{z}).$$

We will denote the fact that $\mathbb{X}$ and $\mathbb{Y}$ are independent given $\mathbb{Z}$ according to the distribution $\mathcal{P}$ as $Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$.

In the case when $\mathbb{Z}$ is an empty set we understand the conditional independence $Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ as the independence of subsets $\mathbb{X}$ and $\mathbb{Y}$, that is the situation when for all $\mathtt{x} \in Dom_{\mathbb{X}}$ and $\mathtt{y} \in Dom_{\mathbb{Y}}$ it holds that:

$$\mathcal{P}(\mathbb{X} = \mathtt{x}, \mathbb{Y} = \mathtt{y}) = \mathcal{P}(\mathbb{X} = \mathtt{x})\mathcal{P}(\mathbb{Y} = \mathtt{y}).$$

The $Ind_{\mathcal{P}}$ notation without any arguments we will use in order to express the set of all conditional independencies existing in the joint probability distribution $\mathcal{P}$. This means that $Ind_{\mathcal{P}}$ is the set of all triples $(\mathbb{X}, \mathbb{Y}, \mathbb{Z})$ of mutually exclusive subsets of $\mathcal{V}$ such that $\mathbb{X}$ and $\mathbb{Y}$ are nonempty and $Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ holds.

In the case when $\mathbb{X}$, $\mathbb{Y}$ or $\mathbb{Z}$ is a one-element set we will use the same $Ind_{\mathcal{P}}$ notation, and for the sake of convenience we will omit brackets in such one-element sets. For example in the case when $\mathbb{X} = \{X\}$ is a one-element set we will denote the conditional independence of $\mathbb{X}$ and $\mathbb{Y}$ given $\mathbb{Z}$ as $Ind_{\mathcal{P}}(X, \mathbb{Y} \mid \mathbb{Z})$ instead of $Ind_{\mathcal{P}}(\{X\}, \mathbb{Y} \mid \mathbb{Z})$

In the purely theoretical setting as the one defined above there is an exact condition which must be satisfied in order to tell that some conditional independence holds. In practice such statement is not so obvious, as we only have some finite sample available. This sample usually does not reflect the generative distribution—assuming it exists—perfectly. As a consequence a decision regarding the existence of some conditional independence given an available dataset is in its nature uncertain. We must accept the situation that some of our decisions will be wrong.

But the mistake rate can be partially controlled. This is one of the fundamental property of statistical tests. They are commonly applied in order to test many different hypothesis about the nature of the generative distribution of a given on the entry sample. In our case we adopt for our purposes statistical tests of conditional independence.

2.2.2. Main Detection Tools: G^2 and χ^2 Tests

One of the most commonly used nowadays tests of conditional independence, in particular in the domain of Bayesian networks, is the G^2 test.

Assume that we have on the entry some dataset $\mathcal{D}$, which is a sample from some unknown for us joint probability distribution $\mathcal{P}$ of the set of random variables $\mathcal{V}$. Let $X, Y \in \mathcal{V}$ be

two different variables and let $\mathbb{Z}$ be some subset of remaining variables in $\mathcal{V}$ (it might be empty). We would like to test on the basis of our available sample $\mathcal{D}$ whether there holds $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$.

For the given $x \in Dom_X$, $y \in Dom_Y$ and $\mathbb{z} \in Dom_{\mathbb{Z}}$ let us denote:

- by $O^{\mathcal{D}}_{xy\mathbb{z}}$ the number of rows in the table $\mathcal{D}$ where $X = x$, $Y = y$ and $\mathbb{Z} = \mathbb{z}$,
- by $O^{\mathcal{D}}_{x\mathbb{z}}$ the number of rows in the table $\mathcal{D}$ where $X = x$ and $\mathbb{Z} = \mathbb{z}$,
- by $O^{\mathcal{D}}_{y\mathbb{z}}$ the number of rows in the table $\mathcal{D}$ where $Y = y$ and $\mathbb{Z} = \mathbb{z}$,
- by $O^{\mathcal{D}}_{\mathbb{z}}$ the number of rows in the table $\mathcal{D}$ where $\mathbb{Z} = \mathbb{z}$.

Then the G^2 statistic is defined as follows:

$$G^2 = 2 \sum_{x \in Dom_X, y \in Dom_Y, \mathbb{z} \in Dom_{\mathbb{Z}}} O^{\mathcal{D}}_{xy\mathbb{z}} \ln \frac{O^{\mathcal{D}}_{xy\mathbb{z}} O^{\mathcal{D}}_{\mathbb{z}}}{O^{\mathcal{D}}_{x\mathbb{z}} O^{\mathcal{D}}_{y\mathbb{z}}}.$$

The null hypothesis for the test is that the considered independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ holds. Parameters $O^{\mathcal{D}}_{xy\mathbb{z}}$ are the observed in the given sample frequencies. The components of the above sum corresponding to the denominator $O^{\mathcal{D}}_{x\mathbb{z}} O^{\mathcal{D}}_{y\mathbb{z}}$ equal to 0 are not counted—we treat all such components as 0.

Let us additionally define the parameters corresponding to the expected frequency of the configuration $\{X = x, Y = y, \mathbb{Z} = \mathbb{z}\}$ under the null hypothesis, that is assuming conditional independence of X and Y given $\mathbb{Z}$:

$$E^{\mathcal{D}}_{xy\mathbb{z}} = O^{\mathcal{D}}_{\mathbb{z}} \frac{O^{\mathcal{D}}_{x\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}} \frac{O^{\mathcal{D}}_{y\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}} = \frac{O^{\mathcal{D}}_{x\mathbb{z}} O^{\mathcal{D}}_{y\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}},$$

Using the expected frequencies we can equivalently say that the G^2 statistics is defined as:

$$G^2 = 2 \sum_{x \in Dom_X, y \in Dom_Y, \mathbb{z} \in Dom_{\mathbb{Z}}} O^{\mathcal{D}}_{xy\mathbb{z}} \ln \frac{O^{\mathcal{D}}_{xy\mathbb{z}}}{E^{\mathcal{D}}_{xy\mathbb{z}}}.$$

In the past the problem with this statistic was the necessity to compute logarithms. As the result the approximation of this statistic was proposed, namely by Karl Pearson (Pearson, 1900). The resultant Pearson's χ^2 test statistic is defined as:

$$\chi^2 = 2 \sum_{x \in Dom_X, y \in Dom_Y, \mathbb{z} \in Dom_{\mathbb{Z}}} \frac{(O^{\mathcal{D}}_{xy\mathbb{z}} - E^{\mathcal{D}}_{xy\mathbb{z}})^2}{E^{\mathcal{D}}_{xy\mathbb{z}}}.$$

This statistic is in fact a second order Taylor expansion of the natural logarithm around 1—which explains the nature of the G^2 statistic approximation. Nowadays calculating logarithms is not a problem and the G^2 became the leading solution.

2.2.3. Time Complexity of G^2 and χ^2 Testing

The time complexity of calculating either the χ^2 or G^2 test statistic corresponding to the examined relation $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ is $\mathcal{O}(mnrs)$, where m is the size of the sample $\mathcal{D}$, $n = |\mathbb{Z}|$, $r = |Dom_X|$ and $s = |Dom_Y|$.

The explanation is that in order to calculate each of the two test statistics it is sufficient to obtain for each appearing in the dataset configuration of values of the set of variables $\mathbb{Z}$ the

information about frequencies of each possible configuration of values of the pair of variables (X, Y). Any configuration of values of $\mathbb{Z}$ which does not occur in the dataset $\mathcal{D}$ simply does not affect both the statistics. We can gather all required information by traversing one time through all m rows of $\mathcal{D}$ and collecting into some data structure, for example a tree, the information about each next appearing configuration of values of the pair (X, Y) corresponding to the given configuration of $\mathbb{Z}$. The scanned from each row of the dataset $\mathcal{D}$ n values corresponding to the configuration of $\mathbb{Z}$ will appear in the tree as the path from the root to the leaf. In the leaf we place the $r \times s$ table, which aim is to store the frequencies of each possible configuration of values of the pair (X, Y)—given the corresponding configuration of $\mathbb{Z}$. Initially such table is filled with zeros, except the one configuration set to 1 corresponding to a dataset row for which the given path to the new leaf was created. During processing of each next row of the dataset either the new path in the tree corresponding to some new observed configuration of values of $\mathbb{Z}$ is created, or the leaf corresponding to some already existing in the tree path is updated.

Clearly traversing through all the dataset and updating the tree has time complexity $\mathcal{O}(mnrs)$. Final calculation of the χ^2 or G^2 statistic on the basis of the obtained tree has time complexity $\mathcal{O}(mrs)$, as in $\mathcal{O}(rs)$ time we will calculate each of the statistic component corresponding to each configuration of values of $\mathbb{Z}$, that is corresponding to each of the leaves of our tree. There are no more than m leaves, as in the worst case there appears in the tree one new leaf for each row of the dataset.

2.2.4. Approximate versus Exact Test Statistics

In the case when the null hypothesis of conditional independence is true then asymptotically with the grow of the sample size both the values of the χ^2 and G^2 statistic are distributed as χ^2 with appropriate number of degrees of freedom. In the case when there are no structural zeros in the considered conditional relation, that is $|Dom_{\mathbb{Z}}| = \prod_{Z \in \mathbb{Z}} |Dom_Z|$ (all assignments of $\mathbb{Z}$ resulting from the possible assignments of the single included variables have nonzero probability) and for all $x \in Dom_X$, $y \in Dom_Y$ and $\mathbb{z} \in Dom_{\mathbb{Z}}$ it holds that $\mathcal{P}(X = x, Y = y, \mathbb{Z} = \mathbb{z}) > 0$, the number of degrees of freedom is equal to:

$$df = (|Dom_X| - 1)(|Dom_Y| - 1) \prod_{Z \in \mathbb{Z}} |Dom_Z|.$$

In the case of finite samples both the χ^2 and G^2 statistic are only approximately distributed with the aforementioned distribution. But this approximation is better in the case of G^2 test—which is its important practical advantage.

There exists an exact statistical test for independence, or more generally conditional independence, where in the case of finite samples a test statistic is distributed exactly with an appropriate χ^2 distribution. It is the Fisher's exact test (Fisher, 1922). Applying this test might be especially important in the case of very small samples, where the approximate tests can deviate very significantly from the correct behavior. However, calculating exact tests statistics is very time expensive, thus in the case of large samples approximate tests are preferred—they can return reasonable results in feasible time.

2.2.5. Interpretation of G^2 Statistic

It can be easily noticed that the G^2 test is in fact a direct measure of conditional association. Namely, assuming that our dataset $\mathcal{D}$ consists of m rows, let us notice that:

$$G^2 = 2 \sum_{x \in Dom_X, y \in Dom_Y, \mathbb{z} \in Dom_{\mathbb{Z}}} O^{\mathcal{D}}_{xy\mathbb{z}} \ln \frac{O^{\mathcal{D}}_{xy\mathbb{z}} O^{\mathcal{D}}_{\mathbb{z}}}{O^{\mathcal{D}}_{x\mathbb{z}} O^{\mathcal{D}}_{y\mathbb{z}}} =$$

$$= 2m \sum_{\mathbb{z} \in Dom_{\mathbb{Z}}} \frac{O^{\mathcal{D}}_{\mathbb{z}}}{m} \sum_{x \in Dom_X, y \in Dom_Y} \frac{O^{\mathcal{D}}_{xy\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}} \ln \frac{\frac{O^{\mathcal{D}}_{xy\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}}}{\frac{O^{\mathcal{D}}_{x\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}} \frac{O^{\mathcal{D}}_{y\mathbb{z}}}{O^{\mathcal{D}}_{\mathbb{z}}}} =$$

$$= 2m \sum_{\mathbb{z} \in Dom_{\mathbb{Z}}} \mathcal{P}_{\mathcal{D}}(\mathbb{Z} = \mathbb{z}) \sum_{x \in Dom_X, y \in Dom_Y} \mathcal{P}_{\mathcal{D}}(X = x, Y = y \mid \mathbb{Z} = \mathbb{z})$$

$$\ln \frac{\mathcal{P}_{\mathcal{D}}(X = x, Y = y \mid \mathbb{Z} = \mathbb{z})}{\mathcal{P}_{\mathcal{D}}(X = x \mid \mathbb{Z} = \mathbb{z}) \mathcal{P}_{\mathcal{D}}(Y = y \mid \mathbb{Z} = \mathbb{z})} = 2m MI_{\mathcal{D}}(X, Y \mid \mathbb{Z}),$$

where the $O^{\mathcal{D}}_{\dots}$ notation we have introduced in Subsection 2.2.2 and $MI_{\mathcal{D}}(X, Y \mid \mathbb{Z})$ is the conditional mutual information of variables X and Y given the vector of variables $\mathbb{Z}$ calculated over the sample $\mathcal{D}$.

The consequence is that the G^2 statistic is always nonnegative and the bigger value we obtain, the larger confidence of dependence of X and Y given $\mathbb{Z}$ we have. In order to decide on the basis of this value whether we reject or not the null hypothesis about the conditional independence of these variables the classical procedure of calculating the p-value and comparing it with the significance level is applied. Let us briefly explain it.

2.2.6. Calculation of p-Value

The p-value for the χ^2 or G^2 test is the probability of obtaining a more radical value (larger, that is indicating bigger conditional dependence) than the one returned by the test statistic. This probability is calculated with respect to the theoretical asymptotic distribution of the statistic, that is with respect to the χ^2 distribution with $(|Dom_X| - 1)(|Dom_Y| - 1) \prod_{Z \in \mathbb{Z}} |Dom_Z|$ degrees of freedom.

If the p-value is smaller than so called significance level of the test then the null hypothesis is rejected, otherwise we assume conditional independence. The common choice for the significance level is 0.05—although depending on the application another, in particular smaller, level might be preferred. The significance level equal to α means that the probability of rejecting the null hypothesis given that it is true is approximately equal to α—as the G^2 and χ^2 statistics are approximately distributed with the reference χ^2 distribution. Thus by means of the significance level we have ability to approximately control at least one type of the error.

Fortunately the type of the error somehow controlled in the considered here tests is a crucial one. It leads to the situation where we are almost always detecting conditional independence whenever it exists. This, as we will see it further, leads in practical application of these tests in Bayesian network structure learning methods to faster detection of sparser graphs.

The smaller is the p-value, the higher is our confidence about the conditional dependence in the considered relation. The p-value can be as a consequence applied itself as a measure of conditional association. Namely, in several Bayesian network structure learning algorithms from the local learning branch, which will be the topic of Chapter 5, as a one of possible choices there is applied the following measure:

$$Assoc_{\mathcal{P}}(X, Y \mid \mathbb{Z}) = 1 - p, \tag{2.2.6.1}$$

where p is the p-value obtained from the performed statistical test investigating the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$.

2.2.7. Adjustment of Degrees of Freedom

In the case when there are structural zeros in the considered dependency relation, that is for some $(x, y, \mathbb{z}) \in Dom_X \times Dom_Y \times \prod_{Z \in \mathbb{Z}} Dom_Z$ it holds that $\mathcal{P}(X = x, Y = y, \mathbb{Z} = \mathbb{z}) = 0$, the situation becomes more complicated. There are in fact no any theoretical results explaining the exact specification of the asymptotic distribution in such case. A typical solution is to proceed with the test by means of the p-value and significance level exactly as it is explained above—but instead of performing calculations with regard to the χ^2 distribution with aforementioned degrees of freedom, the same distribution with appropriately lowered degrees of freedom is applied. We face with the lack of complete theoretical foundations here, thus the reduction of degrees of freedom is only a heuristic.

One of the most common choices is the reduction proposed by Spirtes et al. (2000). It lowers the standard number of degrees of freedom $(|Dom_X| - 1)(|Dom_Y| - 1) \prod_{Z \in \mathbb{Z}} |Dom_Z|$ by 1 for each structural zero, that is for each configuration $(x, y, \mathbb{z}) \in Dom_X \times Dom_Y \times \prod_{Z \in \mathbb{Z}} Dom_Z$ satisfying $\mathcal{P}_{\mathcal{D}}(X = x, Y = y, \mathbb{Z} = \mathbb{z}) = 0$ (notice that we use $\mathcal{P}_{\mathcal{D}}$—thus we calculate the approximation of the requested probability on the basis of the available sample $\mathcal{D}$). Let us call the adjusted in this way number of degrees of freedom as df^S_{adj}.

Another less restrictive reduction was in particular used in the experiments reported by Tsamardinos et al. (2006). In this approach the number of degrees of freedom of the applied in calculations χ^2 distribution is equal to:

$$df^T_{adj} = \prod_{\mathbb{z} \in Dom_{\mathbb{Z}} : \mathcal{P}_{\mathcal{D}}(\mathbb{Z} = \mathbb{z}) > 0} (Dom_{X|\mathbb{Z}=\mathbb{z}} - 1)(Dom_{Y|\mathbb{Z}=\mathbb{z}} - 1)$$

where $Dom_{X|\mathbb{Z}=\mathbb{z}} = |\{x \in Dom_X : \mathcal{P}_{\mathcal{D}}(X = x, \mathbb{Z} = \mathbb{z}) > 0\}|$ and $Dom_{Y|\mathbb{Z}=\mathbb{z}} = |\{y \in Dom_Y : \mathcal{P}_{\mathcal{D}}(Y = y, \mathbb{Z} = \mathbb{z}) > 0\}|$.

Let us notice that in the case when there are no structural zeros in the considered conditional relation the standard number of degrees of freedom df is equal to the adjusted versions df^S_{adj} and df^T_{adj}.

Structural zeros appropriately reduce the values of df^S_{adj} and df^T_{adj}, but it can be also easily noticed that the inequality $df^S_{adj} \leqslant df^T_{adj}$ always holds. This in particular means that the χ^2 or G^2 test with df^T_{adj} degrees of freedom always results in p-values not smaller than in the case of applying df^S_{adj}. The reason is that the larger number of degrees of freedom we consider, the smaller on the whole domain becomes the cumulative distribution function of the χ^2 distribution.

So applying df^S_{adj} results in the conditional independence test rejecting the null hypothesis of independence more often than in in the case of applying df^T_{adj}. As the result, as it will become for us clear in further chapters, Bayesian network learning methods relying on the conditional independence tests—in particular the χ^2 or G^2 test—learns faster and return sparser networks in the case of applying the Tsamardinos et al. (2006) adjustment of degrees of freedom.

2.2.8. Strategy of Neglecting Unreliable Tests

There is one more important aspect related to performing statistical tests for conditional independence. Although we are able to approximately control the one type of the error by

appropriately setting the significance level (with appropriately high chance we will not reject the null hypothesis of conditional independence given that it is true), we must remember that:

- first of all we control this mistake approximately—only asymptotically in the case of the lack of structural zeros we would have here the full control of this mistake,
- secondly, we do not control the second type of the error, that it the risk of not rejecting the null hypothesis given that it is not true.

Moreover, the reliability of the results of the χ^2 or G^2 test is rapidly decaying in the case of growing conditional set $\mathbb{Z}$ in the examined relation $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$. This is a very serious problem, as typically in the case of constraint-based Bayesian network learning algorithms it is not enough to test conditional independence relations corresponding to very small conditional sets. But when we try to perform such more complex tests we might expect that their accuracy will be definitely insufficient.

The intuitive explanation why the reliability of the test of conditional independence decays with the grow of conditional set is very simple. The number of possible configurations of the variables in this conditional set is growing exponentially with the increase of this set, while during the tests we in particular measure the dependence of X and Y given each such configuration. But the sample size is fixed— so on average the amount of data on which we can measure each such conditional dependence is rapidly decaying.

Some solution for this problem has been proposed by Spirtes et al. (2000). The idea is to abort the calculation of a conditional independence test whenever the sample size is insufficiently big in order to deal reliably with the considered conditional dependence relation. Namely, we do not perform the test with the null hypothesis $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ whenever the sample size has size smaller than $c|Dom_X||Dom_Y|\prod_{Z\in\mathbb{Z}}|Dom_Z|$, where c is some a priori chosen by the user constant. Spirtes et al. (2000) suggested to use $c = 5$.

The question must arise here: what happens when in the situation of the insufficient sample size we decide to not perform the test: do we claim that there is a conditional dependence or independence ? The answer is that it might depend on the application, in particular on the Bayesian network learning method where the performing of these tests appears. We will see in this dissertation the algorithms where both of the conventions are applied.

Our choice in the conducted Bayesian network learning experiments described in Chapter 7 is to detect the conditional independence with the G^2 test with adjusted degrees of freedom df^T_{adj}, and to automatically claim that there is a conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ whenever the size of the sample is smaller than $5|Dom_X||Dom_Y|\prod_{Z\in\mathbb{Z}}|Dom_Z|$. Besides other benefits of such choice which we will explain later, one of the reasons of it is that the main comparison appearing in this chapter is with the algorithm proposed by Tsamardinos et al. (2006) and evaluated by the authors basing on exactly such version of the test.

2.2.9. Markov Blankets

Let us finish this section with the brief introduction of one of the fundamental for us concepts in this work, in particular arising on the conditional independence concept—so called Markov blanket. This notion can be defined without in fact any reference to Bayesian networks, but, as we will see it in further chapters, the connection with this field is very significant.

Exploiting the concept of Markov blankets has in particular led to some interesting Bayesian network learning solutions, in particular to the branch of local learning algorithms, which is the main topic of Chapter 5, and which is introduced in Section 4.3. In this section,

namely in Subsection 4.3.1 we will continue with a detailed discussion about the Markov blankets. Here let us only give the definition, as the concept itself without any connections to Bayesian networks will be relatively soon required, namely in Section 3.3.

Definition 2.2.9.1 (Markov blanket, $MB^{\text{distribution}}_{\text{variable}}$ notation) *Let $\mathcal{V}$ be a set of random variables, where $\mathcal{P}$ is its joint probability distribution, and let $T \in \mathcal{V}$ be one of these variables. The Markov blanket of T with regard to the distribution $\mathcal{P}$, which we denote by $MB^{\mathcal{P}}_{T}$, is a subset of variables $\mathbb{Z} \subseteq \mathcal{V} \setminus \{T\}$ satisfying $Ind_{\mathcal{P}}(T, (\mathcal{V} \setminus (\{T\} \cup \mathbb{Z})) \mid \mathbb{Z})$ and such that no proper subset of $\mathbb{Z}$ satisfies this condition.*

It should be pointed out here that the above defined concept was originally (Pearl, 1988) called differently—as a Markov boundary, while the Markov blanket concept was understood like in the above definition—but without the minimality condition. We stick further to the convention given in the above definition—which has become with time most common convention.

The most obvious application of Markov blankets can be considered in isolation from Bayesian networks. Simply the Markov blanket approximation for the decision attribute in some dataset can be used to perform feature selection, which is often a crucial preprocessing step before performing a classification task. Removing from a table all attributes other than these from the Markov blanket of the decision attribute can be justified—in the context of the decision induction all removed variables have no more meaning in a probabilistic sense given the attributes from the blanket of the decision. This selection might be of course not unique, as there can be in general many Markov blankets of some variable.

2.3. Bayesian Network Concept

In this section we finally explain formally what a Bayesian network is. In particular we introduce the main notion on the bases of which Bayesian networks stands—namely the Markov condition. We present the fundamental consequences of this condition, resulting in appropriate properties of Bayesian networks. We illustrate the whole concept on the example. Finally we describe the fundamental graph property related to Bayesian networks, which can be considered as the generalization of Markov condition concept, called d-separation.

2.3.1. Markov Condition

Let us start this section with the main definition.

Definition 2.3.1.1 (Markov condition, Bayesian network) *Let $\mathcal{V}$ be a set of random variables, and let $\mathcal{P}$ be its joint probability distribution. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed acyclic graph (DAG) with vertices corresponding exactly to these random variables. An ordered pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network if it satisfies so called Markov condition: every node (variable) of the graph $\mathcal{G}$ is conditionally independent according to the distribution $\mathcal{P}$ from its nondescendants excluding parents given its parents in this graph.*

This work is focused on the topic of Bayesian network structure learning. Thus the following definition is necessary.

Definition 2.3.1.2 (structure, skeleton, $\pi^{\text{graph}}_{\text{vertex}}$ notation) *By the structure of a Bayesian network $(\mathcal{G}, \mathcal{P})$ we will understand the DAG itself representing this network, that is the graph $\mathcal{G}$.*

By the skeleton of a given graph we will understand the undirected graph resulting from converting each directed edge in the graph into the undirected one.

By the skeleton of a Bayesian network $(\mathcal{G}, \mathcal{P})$ we will mean the skeleton of $\mathcal{G}$, that is the skeleton of the network structure.

The set of parents of a given vertex X in a given graph $\mathcal{G}$ we will denote as $\pi_X^{\mathcal{G}}$.

The following fact is an immediate consequence of the Markov condition. The proof can be found for example in the Neapolitan (2003) book.

Theorem 2.3.1.1 *Let $(\mathcal{G}, \mathcal{P})$ be a Bayesian network, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{V} = \{X_1, ..., X_n\}$. Then the distribution $\mathcal{P}$ can be factored as follows:*

$$\mathcal{P}(X_1, ..., X_n) = \mathcal{P}(X_1 \mid \pi_{X_1}^{\mathcal{G}}) \times ... \times \mathcal{P}(X_n \mid \pi_{X_n}^{\mathcal{G}}). \tag{2.3.1.1}$$

From the given Bayesian network the factorization of the distribution according to the Equation 2.3.1.1 follows. But this works in both directions— any given DAG and factorization built on the basis of this graph lead to the Bayesian network, as it is stated in the following fact. The proof can be also found in the Neapolitan (2003) book.

Theorem 2.3.1.2 *Let $\mathcal{G}$ be a DAG. Let us assign to every node of this graph some conditional probability distribution of this node given every configuration of values of its parents in $\mathcal{G}$. Let us define the joint probability distribution $\mathcal{P}$ basing on the initialized conditional distributions exactly as it is written in Equation 2.3.1.1. Then $(\mathcal{G}, \mathcal{P})$ is a Bayesian network.*

Theorems 2.3.1.1 and 2.3.1.2 lead us to the aforementioned in the introduction practical understanding of the Bayesian network concept. Namely, we can equivalently state basing on the above two facts that the Bayesian network is a DAG, where the vertices represent random variables, equipped with the parameters reflecting the probability distribution of each vertex subject to its parents in the graph. Such structure simply encodes the joint probability distribution from the original definition by means of Equation 2.3.1.1.

According to the Markov condition in the joint probability distribution represented by any Bayesian network each variable is independent from the set of its nondescendants given the set of its parents in this network. That is why the parents of any variable in a Bayesian network can be treated as the direct causes of this variable, as it has been mentioned in the introduction and Subsection 2.1.4.

2.3.2. Sparse Models Preference

Let us notice that a directed clique is able to represent any joint probability distribution in the form of Bayesian network—simply because the Markov condition related to such clique is an empty condition. Every nondescendant of any variable is its parent in such clique, so the independence of the variable from its nondescendants given its parents is automatically satisfied.

But it is definitely worth to search for some sparser representation of the given distribution. The sparser network representing a desired distribution we find, the more likely we will see the minimal set of direct causes for each variable—which is one of the reasons why sparser models are preferred.

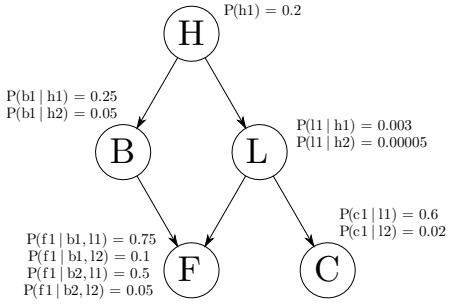
Another reason is that a sparser network is able to represent some joint probability distribution in a more compressed manner. Suppose we have n binary variables. Direct representation of their distribution requires storing $2^n - 1$ numbers, which is exponentially growing

with n. However, if it was possible to represent the joint distribution of these variables with a Bayesian network where each vertex has at most k parents, then such network would require only $n2^k$ parameters (we need to store for each of the n variables only the information about its probability distribution with regard to not more than 2^k possible configurations of values of its parents), which is linearly growing with n, assuming that k can remain constant.

2.3.3. Bayesian Network Example

An exemplary Bayesian network is presented in Figure 2.3.3.1. It corresponds to our dataset and graph examples given in Figures 2.1.1.1 and 2.1.4.1. The parameters of this network reflect the conditional probability of each vertex subject to its parents in the graph. From these parameters we can for example obtain that the joint probability distribution represented by this network—let us name it $\mathcal{P}$—satisfies:

$$\begin{aligned}\mathcal{P}(H = h1, B = b1, L = l2, F = f2, C = c1) = \mathcal{P}(H = h1) \times \mathcal{P}(B = b1 \mid H = h1) \times \\ \times \mathcal{P}(L = l2 \mid H = h1) \times \mathcal{P}(F = f2 \mid B = b1, L = l2) \times \mathcal{P}(C = c1 \mid L = l2) = \\ = 0.2 \times 0.25 \times 0.997 \times 0.9 \times 0.02 = 0.0008973.\end{aligned}$$



Variable	Value	Description
H	$h1$	there is a history of smoking
H	$h2$	there is no history of smoking
B	$b1$	bronchitis is present
B	$b2$	bronchitis is absent
L	$l1$	lung cancer is present
L	$l2$	lung cancer is absent
F	$f1$	fatigue is present
F	$f2$	fatigue is absent
C	$c1$	chest X-ray is positive
C	$c2$	chest X-ray is negative

Figure 2.3.3.1: The network on the left is an exemplary Bayesian network, taken from the Neapolitan (2003) book, illustrating the possible consequences of smoking. The table on the right contains the description of all the considered here variables.

The joint probability distribution represented by some network must satisfy Markov condition with regard to the structure of this network. In particular the joint probability distribution $\mathcal{P}$ of variables H, B, L, F, C corresponding to our exemplary Bayesian network must satisfy the following conditional independencies: $Ind_{\mathcal{P}}(B, \{L, C\} \mid H)$, $Ind_{\mathcal{P}}(L, B \mid H)$, $Ind_{\mathcal{P}}(F, \{H, C\} \mid \{B, L\})$ and $Ind_{\mathcal{P}}(C, \{H, B, F\} \mid L)$.

We can easily get understanding of the factorization property given in Theorem 2.3.1.1 on the basis of our example. Every joint probability distribution, in particular this corresponding to the network from Figure 2.3.3.1, can be factorized according to the chain rule. Assuming that we perform this factorization with regard to the variables order H, B, L, F, C we obtain:

$$\mathcal{P}(H, B, L, F, C) = \mathcal{P}(H) \times \mathcal{P}(B \mid H) \times \mathcal{P}(L \mid H, B) \times \mathcal{P}(F \mid H, B, L) \times \mathcal{P}(C \mid H, B, L, F).$$

Some of the appearing here factors can be now simplified—thanks to this that $\mathcal{P}$ satisfies aforementioned conditional independencies corresponding to the Markov condition. Namely we can reduce the third factor by applying $Ind_{\mathcal{P}}(L, B \mid H)$, the fourth factor by using

$Ind_{\mathcal{P}}(F, \{H, C\} \mid \{B, L\})$ and the last one with $Ind_{\mathcal{P}}(C, \{H, B, F\} \mid L)$. As the result we obtain:

$$\mathcal{P}(H, B, L, F, C) = \mathcal{P}(H) \times \mathcal{P}(B \mid H) \times \mathcal{P}(L \mid H) \times \mathcal{P}(F \mid B, L) \times \mathcal{P}(C \mid L),$$

which is exactly the property given in Equation 2.3.1.1.

2.3.4. d-Separation

As we can see on the basis of the above example the compression of the distribution in Bayesian networks is obtained via exploiting the occurring conditional independencies. The Markov condition corresponding to a given DAG is simply a set of conditional independencies which are necessary in order to perform such compression with respect to this graph.

The Markov condition is defining restrictions about the conditional independencies appearing in any joint probability distribution which can be represented by this graph in the form of Bayesian network. The following definition is related to all such restrictions which can be inferred from this kind of basis.

Definition 2.3.4.1 (entailed conditional independence) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a DAG, where $\mathcal{V}$ is the set of random variables. Let $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subset \mathcal{V}$ be three mutually exclusive subsets, where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty. We say that the graph $\mathcal{G}$ entails based on the Markov condition the conditional independence of $\mathbb{X}$ and $\mathbb{Y}$ given $\mathbb{Z}$ if $Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ holds for every joint probability distribution $\mathcal{P}$ such that $(\mathcal{G}, \mathcal{P})$ satisfies the Markov condition.*

In particular all conditional independencies corresponding to the Markov condition are obviously entailed. It turns out that all entailed by a given DAG conditional independencies can be read directly from its structure, in the form of so called d-separations. Let us introduce this notion.

Definition 2.3.4.2 (chain, blocked chain) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a DAG. Let us consider k different vertices $X_1, ..., X_k \in \mathcal{V}$, where $k \geqslant 2$. We say that the sequence $[X_1, ..., X_k]_{\mathcal{G}}$ is a chain between X_1 and X_k in $\mathcal{G}$ if for all $2 \leqslant j \leqslant k$ it holds that $(X_{j-1}, X_j) \in \mathcal{E}$ or $(X_j, X_{j-1}) \in \mathcal{E}$.*

Let us consider a subset (possibly empty) $\mathbb{Z} \subset \mathcal{V}$, two different vertices $X, Y \in \mathcal{V} \setminus \mathbb{Z}$ and some chain τ between X and Y in $\mathcal{G}$. We say that the chain τ is blocked by $\mathbb{Z}$ in $\mathcal{G}$, if at least one of the following conditions is true:

- *There exists a vertex $Z \in \mathbb{Z}$ which is an element of τ with not converging arrows, that is Z is not simultaneously a children of both of its neighbors in the chain.*
- *There exists a vertex Z which is an element of τ different than X and Y and with converging arrows (Z is a children of both of its neighbors in the chain) such that Z and every descendant of Z does not belong to $\mathbb{Z}$.*

Definition 2.3.4.3 (d-separation, Ind_{graph} notation) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a DAG. Let us consider three mutually exclusive subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subset \mathcal{V}$, where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty. We say that $\mathbb{X}$ and $\mathbb{Y}$ are d-separated by $\mathbb{Z}$ in $\mathcal{G}$, if for every $X \in \mathbb{X}$ and every $Y \in \mathbb{Y}$ each chain between X and Y in $\mathcal{G}$ is blocked by $\mathbb{Z}$ in this graph. We will denote d-separation of $\mathbb{X}$ and $\mathbb{Y}$ given $\mathbb{Z}$ in $\mathcal{G}$ as $Ind_{\mathcal{G}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$.*

The $Ind_{\mathcal{G}}$ notation without any arguments we will interpret as the set of all d-separations occurring in $\mathcal{G}$—that is the set of all triples $(\mathbb{X}, \mathbb{Y}, \mathbb{Z})$ of mutually exclusive subsets of $\mathcal{V}$ such that $\mathbb{X}$ and $\mathbb{Y}$ are nonempty and $Ind_{\mathcal{G}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ occurs.

For example, the DAG presented in Figure 2.3.4.1—let us name it $\mathcal{G}$—contains the following d-separations: $Ind_{\mathcal{G}}(X, \{T, R, S\} \mid \{Y, Z\})$ (the first type of block appears here), $Ind_{\mathcal{G}}(Y, Z \mid X)$ (both the first type of block—in the vertex X, and the second type—in the vertex R or S, appears here), or $Ind_{\mathcal{G}}(W, \{X, Z\} \mid \emptyset)$ (for every possible connecting chain the second type of block appears: in Y, in R or in S), while it does not contain $Ind_{\mathcal{G}}(\{W, T\}, X \mid \emptyset)$ (because the chain $[X, Z, R, T]_{\mathcal{G}}$ is clearly not blocked by the empty set).

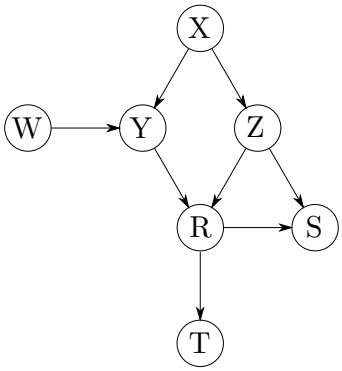


Figure 2.3.4.1: Picture of some DAG taken from the Neapolitan (2003) book, on the basis of which the d-separation concept is exemplified.

For the sake of simplicity we omit the brackets in one-element sets inside the $Ind_{\text{graph}}(...)$ notation—similarly as in the case of the $Ind_{\text{distribution}}(...)$ notation. This will also hold in the word description of the corresponding relation. For example we will say that X (instead of $\{X\}$) and $\{T, R, S\}$ are d-separated by $\{Y, Z\}$ in $\mathcal{G}$.

There is a strong connection between the two defined concepts—conditional independence and d-separation, justifying the analogical notation used for both of them. It can be expressed by the following fact. The proof of this property can be found again for example in the Neapolitan (2003) book.

Theorem 2.3.4.1 *A DAG $\mathcal{G}$ entails based on the Markov condition all and only these conditional independencies which correspond to d-separations in $\mathcal{G}$.*

The above theorem gives us a clear confirmation what kind of distributions can be represented by a given DAG in the form of Bayesian networks. Namely, for a given DAG $\mathcal{G}$, the set of all distributions $\mathcal{P}$ such that $(\mathcal{G}, \mathcal{P})$ satisfies the Markov condition is exactly the set of distributions defined over the nodes of $\mathcal{G}$ which contains all conditional independencies corresponding to d-separations in this graph. In other words, we can formulate the following conclusion.

Corollary 2.3.4.1 *For the given DAG $\mathcal{G}$ and joint probability distribution $\mathcal{P}$ operating on the same set of nodes—random variables the pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network if and only if $Ind_{\mathcal{P}} \supseteq Ind_{\mathcal{G}}$.*

The explanation is as follows. Any distribution $\mathcal{P}$ satisfying d-separations appearing in $\mathcal{G}$ in the form of conditional independencies satisfies as well according to Theorem 2.3.4.1 all entailed by $\mathcal{G}$ based on the Markov condition conditional independencies—so in particular the pair $(\mathcal{G}, \mathcal{P})$ satisfies the Markov condition. In the opposite direction, if we consider any distribution $\mathcal{P}$ such that $(\mathcal{G}, \mathcal{P})$ satisfies the Markov condition, then in particular $\mathcal{P}$ contains

all conditional independencies entailed by $\mathcal{G}$ based on the Markov condition—so according to Theorem 2.3.4.1 it contains as well all conditional independencies corresponding to d-separations in $\mathcal{G}$.

2.4. Constraint-Based Learning

This section is an introduction to one of the two main branches of Bayesian network structure learning methods. This branch is especially important in the context of this work. After some comments about the general scheme of Bayesian network structure learning we introduce the main concept of the constraint-based learning—so called perfect map, which is in a strong sense an optimal structure. Then we investigate the significant patterns characterizing given Bayesian network structure and introduce so called Markov equivalence relation. Finally we state the main theoretical aim of the constraint-based learning—searching for the Markov equivalence class of perfect maps, and we in particular coin the fundamental idea standing behind retrieving the characterizing elements of the desired class with an assist of statistical tests for conditional independence.

2.4.1. Bayesian Network Structure Learning Process

Let us start with a general comment. In the Bayesian network learning process we assume that we have on the entry a dataset $\mathcal{D}$, for which some generative joint probability distribution $\mathcal{P}$ of attributes understood as random variables exists. The process is typically divided into two steps:

- learning the structure of Bayesian network, which in practice approximately represents the generative distribution $\mathcal{P}$,
- estimating the parameters of the network reflecting the conditional distribution of each node given each configuration of values of its parents.

Both steps are in fact very challenging. In the case of the first step there are several well known direct theoretical results which explain its hardness—we will see some of the most notable such statements in Chapter 3. As all this chapter is dedicated to this topic, let us concentrate here a bit more on the second aspect.

One may ask, what is the problem in the second step ? The simple estimation of the parameters based on the conditional frequencies appearing in the data might seem to be an obvious solution. However we must be aware that such thoughtless procedure might lead to several very serious problems in practice. Imagine that we have to assign the probability distribution of some vertex in a really big Bayesian network structure, where this vertex has for example several dozens of parents. The first obvious problem is that both the time and space complexity of such assignment is infeasible in practice—there are simply too many parameters to set. But the second problem is the reliability of these parameters. With the increase of the number of parents each parameter corresponding to this conditional distribution is calculated on the rapidly decreasing sample partition corresponding to the given parents assignment.

These problems open a whole space of research, which is not the main topic of this dissertation, but is indeed also a very interesting field. One can imagine for example that we somehow convert the considered conditional distribution relations into some simpler formulations, where we operate on some ranges or groups of possible values assignments and on the level of such groups only we define the whole probabilistic relation. Moreover, we can even

completely forget about distributions, and formulate all conditional relations in the purely qualitative form, instead of the quantitative. The recognizable model of such form is a qualitative probabilistic network (Wellman, 1990), where instead of any conditional probability distributions we equip the network structure with some qualitative relations only. It is an abstraction of classical Bayesian network model. Inferring from such model does not correspond to calculating any concrete probability of the interesting events given the assignment to some evidence nodes—it rather corresponds to deciding whether the probability of some event grows or decreases given the evidence.

A reader interested in this topic is encouraged to look at the Druzdzel and van der Gaag (1995) work, where in particular several more important problems related to adjusting the parameters of the Bayesian networks and some qualitative versions of them are pointed out, as well as there are proposed some interesting solutions in this field related to appropriate utilization of the domain expert knowledge—which, similarly as the dataset itself is not able to express reliably every conditional relation. Moreover, the number of references to many different Bayesian network models abstractions, ranging from the purely qualitative to purely quantitative ones can be found here.

The most attention of the research in the domain of Bayesian network structure learning is focused on the first step. Note that whatever model of representing relations between attributes we use, the first thing we need is the graphical structure on the basis of which these relations are considered. As we have discussed it in the previous section, the sparser Bayesian network representing given distribution we find, the more valuable is such model. This sparsity of the network is totally dependent on the first step—learning the structure, that is a DAG without parameters—and here lies the main importance of this structure construction process.

2.4.2. Perfect Map

One more not explained formally concept appearing in the introduction is so called perfect map. As we will see it further, it is an optimal Bayesian network structure which can be learned. Constraint-based Bayesian network learning methods are typically aimed at approximating such model.

Definition 2.4.2.1 (perfect map / faithful representation) *We say that a DAG $\mathcal{G}$ is the perfect map of some joint probability distribution $\mathcal{P}$ operating on the set of vertices of $\mathcal{G}$, or that $\mathcal{G}$ is the faithful representation of the distribution $\mathcal{P}$, if it holds that $Ind_{\mathcal{P}} = Ind_{\mathcal{G}}$.*

According to the final conclusions from the previous subsection, namely Corollary 2.3.4.1, the perfect map of a distribution $\mathcal{P}$ is a graph containing the largest possible set of d-separations among all DAGs which can represent $\mathcal{P}$ in the form of Bayesian network. What must be first of all emphasized is that unfortunately a perfect map does not always exist—only specific joint probability distributions have them. In the case when for some distribution there exists a perfect map we say that such distribution admits a faithful representation.

Finding a perfect map is the theoretical aim of many Bayesian network learning algorithms, especially these from the constraint-based family. The underlying idea is that even if the perfect map of a considered distribution does not exist a procedure which in theoretical conditions could lead to this perfect map whenever it exists in practice can still result in finding a high quality Bayesian network.

Some explanation why a perfect map is so much desired gives the following theorem, showing a crucial correlation between the two concepts: a set of d-separations and network sparsity. But first of all let us define one new concept appearing in this theorem.

Definition 2.4.2.2 (covered edge) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, and let us consider $X, Y \in \mathcal{V}$ such that $(X, Y) \in \mathcal{E}$. We say that the edge (X, Y) is a covered edge in $\mathcal{G}$ if*

$$\pi_X^{\mathcal{G}} = \pi_Y^{\mathcal{G}} \setminus \{X\}.$$

So in other words the covered edge is such edge that after removing it from the graph both so far connected vertices have the same set of parents.

Theorem 2.4.2.1 *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be DAGs operating on the same set of vertices. The following conditions are equivalent:*

(a) $Ind_{\mathcal{G}_1} \supseteq Ind_{\mathcal{G}_2}$,

(b) *$\mathcal{G}_1$ can be obtained from $\mathcal{G}_2$ after a sequence of operations, where each operation is removing an edge or reversing a currently covered edge, provided that after each step of the sequence the modified graph remains a DAG.*

The implication from (a) to (b) in this theorem is much harder to prove—it was even a hypothesis for a few years, stated by Meek (1997), and finally proved by Chickering (2002).

An immediate conclusion from Theorem 2.4.2.1 is that if for DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ operating on the same set of vertices it holds that $Ind_{\mathcal{G}_1} \supseteq Ind_{\mathcal{G}_2}$, then the skeleton of $\mathcal{G}_1$ is a subgraph of the skeleton of $\mathcal{G}_2$. This leads us to the conclusion that the perfect map of some joint probability distribution $\mathcal{P}$, assuming it exists, is a DAG with the sparsest possible skeleton among all DAGs which can represent distribution $\mathcal{P}$ in the form of Bayesian network. That is why a perfect map can be undoubtedly considered as a one of the possible interpretations of Bayesian network structure optimality.

2.4.3. Markov Equivalence

Now let us introduce a very natural equivalence relation among DAGs operating on some fixed set of vertices. As it has been outlined in the previous paragraphs the impact of a Bayesian network is all about the amount of d-separations it contains. If two graphs have the same set of d-separations, then it makes sense to treat them as equally powerful. This is exactly the situation which we face in the following definition.

Definition 2.4.3.1 (Markov equivalence) *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be DAGs operating on the same set of vertices $\mathcal{V}$. We say that $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if $Ind_{\mathcal{G}_1} = Ind_{\mathcal{G}_2}$.*

In addition, let us define the following significant Bayesian network structure pattern.

Definition 2.4.3.2 (v-structure) *In a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ each triple of different vertices (X, Z, Y) satisfying $(X, Z) \in \mathcal{E}$, $(Y, Z) \in \mathcal{E}$, $(X, Y) \notin \mathcal{E}$ and $(Y, X) \notin \mathcal{E}$ we call a v-structure.*

For example, the graph presented in Figure 2.3.4.1 contains two v-structures (W, Y, X) and (Y, R, Z).

What might seem to be surprising, the Markov equivalence between two DAGs can be easily verified. Pearl et al. (1990) gave the following characterization of this relation.

Theorem 2.4.3.1 *DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ operating on the same set of vertices are Markov equivalent $\iff$ $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and the same v-structures.*

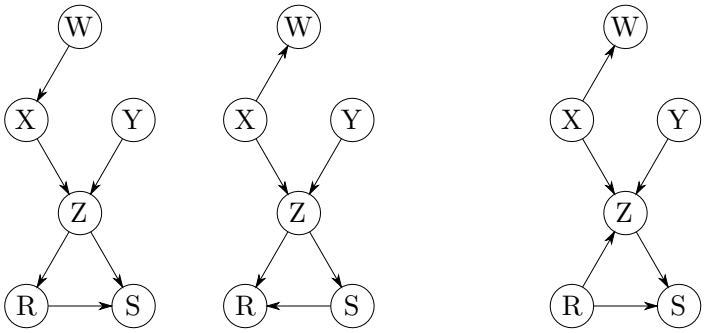


Figure 2.4.3.1: The picture of some DAGs taken from the Neapolitan (2003) book, on the basis of which the Markov equivalence concept is exemplified.

For example let us consider the DAGs illustrated in Figure 2.4.3.1. All three graphs have the same skeleton, but the first two have only one v-structure: (X, Z, Y), while the third graph has two: (X, Z, Y) and (X, Z, R). So thanks to Theorem 2.4.3.1 we know that these first two graphs are Markov equivalent, while the third one is from a different Markov equivalence class.

What is important to notice is that the set of all perfect maps for a given joint probability distribution—if not empty—is exactly a one whole Markov equivalence class. This class, not a particular perfect map, is often a theoretical aim of the Bayesian network learning algorithms from constraint-based family. To identify this class it is sufficient to determine its characterization: the skeleton and the set of v-structures. Any DAG consistent with this characterization is an exemplary perfect map.

2.4.4. Foundations of Constraint-Based Learning

One of the fundamental constraint-based methodologies of Bayesian network induction, the SGS algorithm (Spirtes et al., 1990), together with its enhancement—the PC algorithm (Spirtes and Glymour, 1991), is based in fact on the two simple observations about the nature of d-separations. These observations turned out to be the crucial properties used to justify many later constraint-based algorithms. The correctness of the presented in this dissertation procedure of Bayesian network induction from local structures is proven using in particular also these facts.

Remark 2.4.4.1 *Let us consider a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and two different vertices $X, Y \in \mathcal{V}$. Then it holds that: X and Y are connected by an edge in $\mathcal{G} \iff X$ and Y are not d-separated by any subset of the vertices from $\mathcal{V} \setminus \{X, Y\}$.*

Remark 2.4.4.2 *Let us consider a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and three different vertices $X, Y, Z \in \mathcal{V}$, such that X and Y are not connected by an edge in $\mathcal{G}$, while the pair X and Z and the pair Y and Z is connected. Then:*

- *If the triple (X, Z, Y) is a v-structure, then X and Y are d-separated by some subset of the remaining vertices excluding Z, and are not d-separated by any subset of the remaining vertices containing Z.*
- *If the triple (X, Z, Y) is not a v-structure, then X and Y are d-separated by some subset of the remaining vertices containing Z, and are not d-separated by any subset of the remaining vertices excluding Z.*

Remark 2.4.4.2 is an instant consequence of Remark 2.4.4.1 and the d-separation definition. In Remark 2.4.4.1 the implication $\Longrightarrow$ is obvious, while the implication $\Longleftarrow$ can be derived from the following observation.

Remark 2.4.4.3 *Let us consider a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and two different vertices $X, Y \in \mathcal{V}$, such that X and Y are not connected by an edge in $\mathcal{G}$. Then it holds that X and Y are d-separated by at least one of the vertices subsets: $\pi_X^{\mathcal{G}}$ or $\pi_Y^{\mathcal{G}}$.*

The above described facts and their simple proofs can be found for example in the Neapolitan (2003) book.

Remarks 2.4.4.1 and 2.4.4.2 can be used directly to learn the Markov equivalence class of the perfect maps of some distribution, assuming it admits a faithful representation. All we need is the knowledge about the conditional independencies appearing in the considered distribution, as these independencies reflect all d-separations occurring in the desired faithful representation. The knowledge about all these d-separations can be used then in order to determine with an assist of the remarks the skeleton and the set of v-structures characterizing each DAG containing exactly this set of d-separations—that is we can determine the representation of the desired class of perfect maps.

In practice of course there are numerous problems appearing. First of all the complexity of finding all d-separations is overwhelming—only some relatively small part of the conditional independencies over considered variables can be usually examined. Secondly, the acquired knowledge about the conditional independencies is imperfect—statistical tests for conditional independence are calculated over some available finite sample and the tests results can be wrong. The problem is even becoming bigger when the conditioned set is big for considered conditional independence. In such a case reliability of the tests results can be drastically decreased. The described in Subsection 2.2.8 Spirtes et al. (2000) solution of neglecting the least reliable tests might be insufficient. As a consequence what is often desired from the constraint-based Bayesian network learning methods is to obtain—subject to some simplifying assumptions (for example some kind of sparsity of a desired faithful representation)—a sufficient knowledge from a relatively small amount of reliable statistical tests for conditional independence.

The topic of the constraint-based Bayesian network structure learning algorithms is continued in much more detail in Chapters 4 and 5, where the enormous development of this branch of algorithms is presented.

Chapter 3

Hardness of Bayesian Networks

Bayesian networks provide great possibilities of representing the entire joint probability distributions, giving the opportunity to perform the classification with regard to different conditional attributes and different decision attributes. More generally, any aspect of the joint probability distribution encoded in such network can be extracted—which is a huge advantage over the other types of classification models. Moreover, Bayesian networks are not a black box tools, they provide an intuitive and understandable graphical representation, which can be itself very useful for further analysis.

Bayesian networks provide without doubts a very ambitious knowledge representation. However, the variety of possible applications of them has its computational cost. In the case when we are interested in the exact inferring from the networks, or exact construction of optimal networks from an input data, we must be aware that we face the NP-hard problems.

This chapter is presenting the most fundamental theoretical results describing the hardness of Bayesian networks in such aspects like inferring from the networks and constructing them. By the way we introduce here two classical Monte Carlo approaches aimed at the approximate reasoning from Bayesian networks. Some of the techniques described here have been applied in the experiments reported in Chapter 7. We also describe here briefly the second main class of Bayesian network structure learning algorithms—the class of score-based learning methods. The reason is that the most recognizable theoretical results describing the hardness of Bayesian network structure learning are settled exactly in the scenario of score-based learning. Finally we present a new insight into the hardness of Bayesian network structure learning—a new theoretical result reflecting this hardness in a much easier to understand way than the classical view.

3.1. Inference Aspects

Bayesian networks can be in particular considered as classifiers—even a very powerful ones in the sense that they do not only provide ability to classify one chosen attribute, but classify any of them, and answer the number of other more general queries. All of this is thanks to the fact that Bayesian networks encode the whole joint probability distribution of its attributes. The main problem however is that such inference is in general NP-hard. In this section we explain what we mean by inferring from Bayesian networks, and how looks the classical view on its NP-hardness. We also describe two fundamental approximate inference algorithms. One of them, called logic sampling, will be important for us in Chapter 7, where we apply the technique of samples generation included in this method.

3.1.1. Problem Statement

We will deal in this section with the following problem.

Bayesian-Network-Inference

On the entry we have a Bayesian network $(\mathcal{G}, \mathcal{P})$ represented by the DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ equipped with parameters describing the conditional distribution of each vertex given its parents, where the set of vertices—random variables $\mathcal{V} = \{X_1, ..., X_n\}$. There is also given a subset of indexes $I = \{i_1, ..., i_k\} \subset \{1, ..., n\}$ and $J = \{j_1, ..., j_l\} \subseteq \{1, ..., n\} \setminus I$ where $J \neq \emptyset$, as well as the values: $x_{i_1} \in Dom_{X_{i_1}}, ..., x_{i_k} \in Dom_{X_{i_k}}, x_{j_1} \in Dom_{X_{j_1}}, ..., x_{j_l} \in Dom_{X_{j_l}}$. The aim is to return:

$$\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}).$$

Example 3.1.1.1 *An example problem can be as follows. Assume that on the entry we have some Bayesian network $(\mathcal{G}, \mathcal{P})$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $V = \{X_1, ..., X_6\}$ are binary random variables. The aim is to return:*

$$\mathcal{P}(X_1 = 1, X_2 = 0 \mid X_3 = 0, X_4 = 1).$$

The most fundamental and fast method of extracting an information about the represented joint probability distribution from Bayesian networks is the one corresponding to Equation 2.3.1.1. Namely, we are able to easily extract the information about the probability of any configuration of values of all the random variables—simply by multiplying the appropriate parameters of the network.

In our example in order to calculate the requested probability it is sufficient to obtain the values $\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1)$, $\mathcal{P}(X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1)$, $\mathcal{P}(X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1)$ and $\mathcal{P}(X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 1)$, as:

$$\begin{aligned}\mathcal{P}(X_1 = 1, X_2 = 0 \mid X_3 = 0, X_4 = 1) = \mathcal{P}(X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1)/\\ /[\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1) + \mathcal{P}(X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1)+\\ +\mathcal{P}(X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1) + \mathcal{P}(X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 1)].\end{aligned}$$

Each of these four probabilities can be translated into the sum of atomic probabilities corresponding to configurations of values of all six variables. For example:

$$\begin{aligned}\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1) = \mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 0, X_6 = 0)+\\ +\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 0, X_6 = 1)+\\ +\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 1, X_6 = 0)+\\ +\mathcal{P}(X_1 = 0, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 1, X_6 = 1).\end{aligned}$$

Each of the components of this sum we directly obtain from the network via Equation 2.3.1.1. The three remaining probabilities related to variables X_1, X_2, X_3 and X_4 can be calculated in the analogical fashion.

3.1.2. Inference Hardness

The problem with the above straightforward procedure is its lack of efficiency. In the general case of n binary variables and the query about some conditional probability among these

variables where there are k conditional variables we would have to decompose the problem into the calculation of 2^{n-k} atomic probabilities.

There is a number of classical exact and approximate algorithms performing the inference from Bayesian networks. The most recognizable is a belief propagation algorithm proposed by Judea Pearl. The first version of this algorithm was designed for the case of Bayesian network structures in the form of trees (Pearl, 1982), then it was extended to polytrees (Kim and Pearl, 1983). These are the polynomial in time procedures, returning exact solutions. Namely, both these algorithms return the exact probability $\mathcal{P}(Y = y \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ for all variables Y in the network and all their values y in $\mathcal{O}(nk^p)$ time, where n is the number of variables in the network, k is the maximal number of possible values among these variables, and p is the maximal in-degree. In particular these procedures can be applied to calculate in pessimistically quadratic with regard to n time the requested conditional probability in the considered here more general form, that is $\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ by applying the chain rule:

$$\begin{aligned}
&\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}) = \\
&= \mathcal{P}(X_{j_1} = x_{j_1} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}) \times \\
&\times \mathcal{P}(X_{j_2} = x_{j_2} \mid X_{j_1} = x_{j_1}, X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}) \times \\
&\times \mathcal{P}(X_{j_3} = x_{j_3} \mid X_{j_1} = x_{j_1}, X_{j_2} = x_{j_2}, X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}) \times \\
&\times ... \times \mathcal{P}(X_{j_l} = x_{j_l} \mid X_{j_1} = x_{j_1}, X_{j_2} = x_{j_2}, ..., X_{j_{l-1}} = x_{j_{l-1}}, X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}),
\end{aligned}$$

and calculating each of the factors (there are no more than n of them) using the belief propagation algorithm.

There has also arose on the basis of the mentioned above algorithms an approximate version of the belief propagation algorithm for the case of general DAGs (Pearl, 1988).

There does not exist an exact algorithm performing the inference from a general Bayesian network in a polynomial time with regard to its size. There exist such algorithms only for the case of appropriately sparse Bayesian network structures—for example the mentioned belief propagation. Provided that in the computational complexity theory the hypothesis $P \neq NP$ is true there is no hope to find an efficient algorithm in the general case. The formal argumentation for this claim has been given by Cooper (1990). He has proved that the following decision problem is NP-hard.

Bayesian-Network-Inference—Simple-Decision-Version

Given on the entry a Bayesian network $(\mathcal{G}, \mathcal{P})$ represented by the DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ equipped with parameters describing the conditional distribution of each vertex given its parents, the variable $X \in \mathcal{V}$ and the value $x \in Dom_X$, determine whether $\mathcal{P}(X = x) > 0$.

This in particular means that the more general problem Bayesian-Network-Inference is NP-hard.

However, on the appropriate high level of abstraction the process of inferring can be easy. Let us recall the mentioned in Subsection 2.4.1 notion of qualitative probabilistic network, which is an abstraction of the classical Bayesian network. Here the reasoning corresponds to expressing in the purely qualitative form our change of beliefs regarding the particular attributes given the evidence. We are not going into any formal details, an interested reader can study the Wellman (1990) work. But it is worth to mention that there have been designed time efficient algorithms performing the inference in the qualitative belief networks without any restrictions regarding the sparseness of their structures. In particular there have been proposed algorithms inspired by the belief propagation mechanism performing this inference,

first for the case of the polytrees (Henrion and Druzdzel, 1990), and then also for the case of any network structures (Druzdzel and Henrion, 1993)—both polynomial in time.

In the specific applications the reasoning abilities of qualitative probabilistic networks are more than enough. The rough information regarding the change of beliefs given evidence can be sufficiently valuable comparing to the exact quantitative information. In the context of Bayesian network inference hardness it is thus worth to keep in mind that there exists such alternative and easier to handle models.

3.1.3. Reasoning by Sampling—Logic Sampling

Let us now present two most recognizable Monte Carlo algorithms for the problem of the inference from Bayesian networks. The first algorithm, proposed by (Henrion, 1986) and called logic sampling, is in fact a very straightforward Monte Carlo approach. The idea is following—let us generate a sample from the generative distribution $\mathcal{P}$ represented by the given Bayesian network. Then in order to approximately determine the interesting for us conditional probability corresponding to $\mathcal{P}$ we simply estimate it on the basis of appropriate frequencies occurring in the generated sample.

We are facing here exactly the problem Bayesian-Network-Inference. So in particular we have on the entry a Bayesian network $(\mathcal{G}, \mathcal{P})$ represented by the DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ equipped with parameters where $\mathcal{V} = \{X_1, ..., X_n\}$, and what we want to obtain is:

$$\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k}).$$

In the algorithm there appears one new notion, so we have to first define it.

Definition 3.1.3.1 (ancestral order) *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph, where the set of vertices $\mathcal{V} = \{X_1, ..., X_n\}$. The ancestral order of the vertices in $\mathcal{G}$ is an ordered sequence of the vertices $[X_{a_1}, ..., X_{a_n}]$, corresponding to some permutation of all the vertices included in $\mathcal{V}$, such that for each $1 \leqslant i < j \leqslant n$ there is no edge in $\mathcal{G}$ from the vertex X_{a_j} to X_{a_i}.*

Description

For directed acyclic graphs there always exists an ancestral order of vertices. Algorithm 3.1.3.1—the logic sampling algorithm solving approximately our problem—is based on this property. The algorithm estimates the requested conditional probability on the basis of the generated sample of size m. This m can be either the parameter specified a priori by the user, or there can be done some automatic assignment basing on some typical conventions. The generation of each next row of the sample is performed in the ancestral order of the vertices in the network structure.

The step 2 is probably the one requiring some more explanation. The notation $D[...]$ appearing here was explained in Subsection 2.1.1. We can however express in the words how the process of generating each next row of the sample is performed. For each next attribute proceeded in the ancestral order of the nodes in the structure of the input Bayesian network we generate randomly and independently the value of this attribute with regard to the conditional distribution of it given the already assigned configuration of values of the attributes—parents in the graph. The parents have already assigned values, because we assign the values to attributes sorted in the ancestral order. Moreover the considered conditional distribution can be instantly derived from the Bayesian network—in the form of the parameters assigned to the currently processed variable.

Algorithm 3.1.3.1 The logic sampling algorithm.

1. Determine the ancestral order $[X_{a_1}, ..., X_{a_n}]$ of the vertices $X_1, ..., X_n$ in the structure $\mathcal{G}$ of the input Bayesian network $(\mathcal{G}, \mathcal{P})$.

2. Let us generate the sample from the distribution $\mathcal{P}$ of size m, that is the dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$, where $\mathcal{U} = \{U_1, ..., U_m\}$, $\mathcal{V} = \{X_1, ..., X_n\}$. Namely, the attributes values for each object $U \in \mathcal{U}$ are assigned in the following way:

   - For i from 1 to n set randomly and independently the value $\mathcal{D}[U, X_{a_i}]$ with regard to the conditional distribution $\mathcal{P}(X_{a_i} \mid \pi^{\mathcal{G}}_{X_{a_i}} = \mathcal{D}[U, \pi^{\mathcal{G}}_{X_{a_i}}])$.

3. Return the estimation of the value $\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ calculated as $\mathcal{P}_{\mathcal{D}}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$.

The sign $\mathcal{P}_{\mathcal{D}}$ appearing in Step 3 was introduced in Subsection 2.1.2. Recall that it expresses the joint probability distribution directly derived from the frequencies of values configurations of the attributes in the dataset $\mathcal{D}$.

The generated auxiliary dataset in the logic sampling algorithm is indeed a sample from the generative distribution $\mathcal{P}$ represented by the input Bayesian network $(\mathcal{G}, \mathcal{P})$—the sample understood strictly according to Definition 2.1.3.1. Let us notice that each next row is indeed independently generated from distribution $\mathcal{P}$. Recall Equation 2.3.1.1—the probability of obtaining any configuration of values of each next generated row of the sample is clearly directly corresponding to the factorized representation given in this equation—as the value of each next processed attribute is generated independently on the other generated values and with regard to its conditional distribution given the already chosen values assignment of its parents in the network structure.

Time Complexity

Let us analyze now the time complexity of the logic sampling algorithm with regard to the size of the structure of the input Bayesian network expressed by the number of nodes $n = |\mathcal{V}|$ and number of edges $l = |\mathcal{E}|$, and with regard to m—the size of the generated auxiliary sample. First of all we have to determine the ancestral order of the variables in the network structure. We can do it in $\mathcal{O}(n+l)$ time, by applying the depth-first search algorithm. Then the process of generating the sample starts. We generate m rows, and the generation of each row costs us pessimistically $\mathcal{O}(n^2)$ time—as pessimistically processed vertices can have $\mathcal{O}(n)$ parents (namely in order to perform the random choice of the attribute value for each vertex we need to gather the appropriate parameters from the network corresponding to the configuration of values of its parents, which costs us $\mathcal{O}(n)$ time). In order to perform in Step 3 the estimation of the requested probability we have to pessimistically traverse through all the dataset once in order to calculate the numerator in the estimation, that is the number of objects U satisfying:

$$\{\mathcal{D}[U, X_{j_1}] = x_{j_1}, ..., \mathcal{D}[U, X_{j_l}] = x_{j_l}, \mathcal{D}[U, X_{i_1}] = x_{i_1}, ..., \mathcal{D}[U, X_{i_k}] = x_{i_k}\}$$

and the denominator, that is the number of objects U satisfying:

$$\{\mathcal{D}[U, X_{i_1}] = x_{i_1}, ..., \mathcal{D}[U, X_{i_k}] = x_{i_k}\}.$$

This traverse costs us $\mathcal{O}(mn)$ time.

The cost $\mathcal{O}(n+l)$ of Step 1 and the cost $\mathcal{O}(mn)$ of Step 3 are entailed in the cost of Step 2: $\mathcal{O}(mn^2)$. So the total time complexity of the logic sampling algorithm is $\mathcal{O}(mn^2)$.

In the special cases of an appropriately sparse class of the Bayesian networks on the input this algorithm becomes even more efficient. For example, if we assume that each of the vertices in the network structure has at most k parents the algorithm has the time complexity $\mathcal{O}(mn)$—as Step 1 costs here only $\mathcal{O}(n)$ time (there are only $\mathcal{O}(n)$ edges in the graph), while Step 2 has in such case the complexity reduced to $\mathcal{O}(mn)$ (for each generated cell of the dataset we need to process the configuration of values of the parents of corresponding vertex in the network structure, which has here bounded by a constant size).

It must be pointed out that the logic sampling has some very serious drawback in the context of its main application, that is the reasoning from Bayesian networks. Notice that the estimation of the requested conditional probability $\mathcal{P}_{\mathcal{D}}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ uses a potentially very small portion of the whole generated auxiliary dataset corresponding to only these objects U which satisfy $\{\mathcal{D}[U, X_{i_1}] = x_{i_1}, ..., \mathcal{D}[U, X_{i_k}] = x_{i_k}\}$. Imagine that this condition corresponds even to the simplest binary attributes. With the grow of this condition, that is the value k, the indicated configuration of values becomes on average rapidly very infrequent in the dataset—as there are 2^k possible configurations of this conditional variables. This means that either we need to rapidly increase the size of the generated sample with the grow of k, or we must accept that our estimation will be based in fact on the unrealistically small subsample corresponding to the specified condition—which results in a very unreliable estimation.

3.1.4. Reasoning by Sampling—Likelihood Weighting

On the basis of the idea on which the logic sampling method is based there has arose several much more sophisticated algorithms. One of the most recognizable and significant enhancement of the logic sampling algorithm is called likelihood weighting (Fung and Chang, 1989).

Description

We have presented the method in Algorithm 3.1.4.1. The fundamental difference here is placed in fact in the one crucial aspect—we generate here the sample from the input Bayesian network, where each of the generated rows U satisfies $\{\mathcal{D}[U, X_{i_1}] = x_{i_1}, ..., \mathcal{D}[U, X_{i_k}] = x_{i_k}\}$. The estimation of the requested conditional probability is thanks to this based on all the generated sample, not the potentially very small portion of it—which means that the reliability of the estimation is for some fixed size of the generated sample not decreasing with the increase of k— the size of the set of conditional variables.

As it can be noticed in addition to the dataset $\mathcal{D}$ we generate an assignment of each created object of $\mathcal{D}$ to some real positive number. This assignment can be interpreted as a weighting function—each of the rows of the sample obtains such weight, which is then used in order to estimate the requested conditional probability.

Namely, in Step 3 the conditional probability we estimate with regard to the distribution $\mathcal{P}^w_{\mathcal{D}}$. We must explain what in fact this distribution is. Similarly as the introduced in Subsection 2.1.2 distribution $\mathcal{P}_{\mathcal{D}}$ the distribution $\mathcal{P}^w_{\mathcal{D}}$ is reflecting directly the frequencies appearing in the dataset $\mathcal{D}$—but with additional respect to the weights assigned to the rows.

Namely, for the given dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$, any subset of attributes $\mathbb{X} \subseteq \mathcal{V}$ and any values assignment $\mathbb{x}$ of the set of attributes $\mathbb{X}$ the joint probability distribution $\mathcal{P}^w_{\mathcal{D}}$ is defined as:

$$\mathcal{P}^w_{\mathcal{D}}(\mathbb{X} = \mathbb{x}) = \frac{\sum_{U \in \mathcal{U}: \mathcal{D}[U,\mathbb{X}] = \mathbb{x}} w(U)}{\sum_{U \in \mathcal{U}} w(U)}.$$

Algorithm 3.1.4.1 The likelihood weighting algorithm.

1. Determine the ancestral order $[X_{a_1}, ..., X_{a_n}]$ of the vertices $X_1, ..., X_n$ in the structure $\mathcal{G}$ of the input Bayesian network $(\mathcal{G}, \mathcal{P})$.

2. Let us generate the sample of size m with attributes corresponding to variables in the network, that is the dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$, where $\mathcal{U} = \{U_1, ..., U_m\}$, $\mathcal{V} = \{X_1, ..., X_n\}$, together with the additional assignment $w : \mathcal{U} \mapsto [0, \infty)$. Namely, for each object $U \in \mathcal{U}$ the attributes values and the value $w(\mathcal{U})$ are assigned in the following way:

   - Set the values: $\mathcal{D}[U, X_{i_1}] := x_{i_1}, ..., \mathcal{D}[U, X_{i_k}] := x_{i_k}$.
   - For i from 1 to n, if $\mathcal{D}[U, X_{a_i}]$ has not been already filled in the previous point, set the value $\mathcal{D}[U, X_{a_i}]$ randomly and independently with regard to the conditional distribution $\mathcal{P}(X_{a_i} \mid \pi^{\mathcal{G}}_{X_{a_i}} = \mathcal{D}[U, \pi^{\mathcal{G}}_{X_{a_i}}])$.
   - Assign $w(U) = \prod_{i \in \{i_1, ..., i_k\}} \mathcal{P}(X_i = x_i \mid \pi^{\mathcal{G}}_{X_i} = \mathcal{D}[U, \pi^{\mathcal{G}}_{X_i}])$

3. Return the estimation of the value $\mathcal{P}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$ calculated as $\mathcal{P}^w_{\mathcal{D}}(X_{j_1} = x_{j_1}, ..., X_{j_l} = x_{j_l} \mid X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$.

In particular, when to each object we assign the weight equal to 1, distributions $\mathcal{P}_{\mathcal{D}}$ and $\mathcal{P}^w_{\mathcal{D}}$ are exactly the same.

Weighting Reason

The explanation why in the likelihood weighting algorithm we assign weight to each row in order to estimate the requested conditional probability is that this assures an unbiased estimation. Notice that in the joint probability distribution $\mathcal{P}$ represented by the input Bayesian network the probability of each generated row in the likelihood weighting procedure is not the same as the probability of occurrence of such row in the process of generation.

In the process of generation of each next row the variables $X_{i_1}, ..., X_{i_k}$ are already instantiated, so the probability of generating each object U corresponds to the product

$$\prod_{i \in \{1,...,n\} \setminus \{i_1,...,i_k\}} \mathcal{P}(X_i = \mathcal{D}[U, X_i] \mid \pi^{\mathcal{G}}_{X_i} = \mathcal{D}[U, \pi^{\mathcal{G}}_{X_i}]).$$

In contrary, the probability of any object U according to the distribution $\mathcal{P}$ represented by the input network is equal to:

$$\prod_{i \in \{1,...,n\}} \mathcal{P}(X_i = \mathcal{D}[U, X_i] \mid \pi^{\mathcal{G}}_{X_i} = \mathcal{D}[U, \pi^{\mathcal{G}}_{X_i}]).$$

As we can see the weight $w(U)$ represents exactly the product or the missing factors in the generation probability. As the result the corresponding bias with regard to the joint probability represented by the network is removed, and the estimation of the requested conditional probability basing on these weights is unbiased too.

Time complexity

The time complexity with regard to the size m of the generated sample and with regard to the number of vertices n and number of edges l of the input Bayesian network of likelihood

weighting can be summarized as follows. The step 1 is the same as in the case of the logic sampling—so the time complexity remains $\mathcal{O}(n + l)$. In Step 2 although comparing to the logic sampling we do not have to generate the values for attributes $X_{i_1}, ..., X_{i_k}$ as they are automatically assigned to $x_{i_1}, ..., x_{i_k}$, this missing cost is directly shifted to the process of calculating the weight for each object. So Step 2 costs us $\mathcal{O}(mn^2)$ time. The step 3 requires similarly as in the logic sampling one pass through all the dataset, so it costs us $\mathcal{O}(mn)$ time. Here in order to calculate the numerator and denominator of the estimation of the requested conditional probabilities we summarize appropriate weights retrieved during the pass.

As we see the total time complexity of likelihood weighting is $\mathcal{O}(mn^2)$—exactly the same as in the case of the logic sampling. Also in the case of the simplified scenario, where on the input we consider only the network structures having no more than k parents of each node where k is some constant, the complexity of the logic sampling and likelihood weighting are both $\mathcal{O}(mn)$, as in both methods the cost of Step 1 is reduced to $\mathcal{O}(n)$ and the cost of Step 2 is reduced to $\mathcal{O}(mn)$.

3.1.5. Reasoning by Sampling—Conclusions and Further Progress

In practice we must remember that in order to provide with the assist of logic sampling the similarly reliable estimation comparing to likelihood weighting we have to in contrary to this method increase the size of the generated sample with the increase of the number of conditional variables in the requested conditional probability query. Thus in practice logic sampling is undoubtedly less efficient than likelihood weighting.

However, the logic sampling subroutine of generating the auxiliary dataset is itself a very useful tool. It is a simple, efficient and theoretically correct procedure of generating samples from the given reference Bayesian network. Such sample can be then treated as a training dataset—any Bayesian network structure learning algorithm can be then applied on it and the quality of the obtained model with regard to the original model—dataset generator can be easily retrieved. We will discuss this topic in more details in the experiments part of this work, namely in Chapter 7.

It might seem at the first glance that the likelihood weighting algorithm is an ultimate solution. Unfortunately it has also some drawbacks.

Whenever the probability of the evidence, that is $\mathcal{P}(X_{i_1} = x_{i_1}, ..., X_{i_k} = x_{i_k})$, is very high similarly efficient in terms of obtaining analogical precision of the results in the given time can be both logical sampling and likelihood weighting. Although in the case of the small probability of the evidence, which may occur when there are many evidence nodes, corresponding for example to some large network, likelihood weighting typically outperforms logical sampling, the behavior of the winner is not so perfect.

Note that there is one important here difference between logical sampling and likelihood weighting. In the first method each of the generated, let us say m, rows of the sample corresponding to the evidence condition obtains the same weight, equal to 1. In the second method each of the generated rows, assume that their number is also m (all of them corresponds already to the evidence condition), can obtain an arbitrary weight between 0 and 1. After the process of generating the sample the weights are normalized—in the case of logical sampling with regard to m—the number of rows satisfying the evidence condition, while in the case of likelihood weighting with regard to the sum of weights of all m generated rows. In both cases the sum of such normalized weights corresponding to appropriate rows determines the approximation of the requested conditional probabilities. The crucial point is that each of the m normalized weights in the case of logical sample is equal to $\frac{1}{m}$, while in the case of likelihood weighting the m normalized weights are arbitrary positive real numbers summing

to 1.

So the problem is that although in the case of likelihood weighting we eliminate the necessity of generation possibly much bigger irrelevant sample part corresponding to the unsatisfied evidence condition, the generated sample on the basis of which probability approximations are derived is in contrary to logical sampling equipped with an additional level of uncertainty—the variance of weights. This variance can extend significantly the size of the sample required to achieve some satisfactory level of convergence of the requested probability approximations.

One of the directions of the further development in the Monte Carlo Bayesian network inference methods was to reduce at least partially the above sketched problem. The most recognizable is the extension of the likelihood weighting mechanism called self-importance weighting (Shachter and Peot, 1989). The basic idea here is to appropriately modify with time the generative distribution according to which each next row is generated—modify on the basis of so far generated data with the aim of speeding up the convergence rate. Regardless of the interesting properties of such extended mechanism the most popular sample generation-based reasoning method is without doubts likelihood weighting. It is a conceptually simple approach, which is not having especially good convergence rate with regard to the size of the generated sample in general comparing to other approaches like the mentioned self-importance weighting, but thanks to its simplicity it is able to generate in the same time much bigger sample, which results in not significantly lower comparing to these more advanced approaches convergence rate with regard to sample generation time.

Regardless of the popularity of likelihood weighting it should be pointed out that there have been designed with time significantly better and more sophisticated mechanisms, which can easily outperform this method in the hardest case of low probability evidences with respect to the convergence rate both with regard to the size of the generated sample and the time of the sample generation. The method which achieves such properties, called adaptive importance sampling, has been proposed by Cheng and Druzdzel (2000). It is inspired by the self-importance weighting mechanism. The source of the success here is the significantly improved procedure of sampling which has enabled to reach incomparably better conversion rate even with regard to the sample generation time.

3.2. Learning Aspects

We focus here on the second main class of Bayesian network structure learning methods—the class of score-based learning algorithms. We formalize the aim of this learning as searching for so called parameter optimal map, which is a special and stronger case of the introduced in the following subsection so called inclusion optimal map. We explain what the main tool guiding this search—scoring criterion, is, and we show the most desired its properties. Finally we define the classical problems of finite-sample and asymptotic Bayesian network structure learning—both expressed in the score-based paradigm. As we point it out both these problems are NP-hard. Nowadays the classical view on the NP-hardness of Bayesian network structure learning is expressed by meas of especially these two problems.

3.2.1. Inclusion Optimal Map

Not only the inference is in the case of Bayesian networks domain a challenge from the computational point of view. The real problem begins much earlier—on the stage of learning such models. In order to explain this we have to first introduce the second main branch of Bayesian network learning algorithms, the class of score-based learning methods, as the

most significant and notable theoretical results describing the hardness of Bayesian networks learning are settled in the scenario of such algorithms.

Let us recall that in the case of the constraint-based learning methods the usual theoretical aim is to find so called perfect map of the generative distribution of the input dataset—assuming that such distribution exists and that it admits a faithful representation. In the case of the score-based methods the assumptions are less restrictive, and as a consequence the theoretical aim has a more general character. Namely we assume here only that there exists the generative distribution for the given on the input dataset—but we do not assume the existence of any perfect map for this distribution. In such a case some another theoretical aim of the Bayesian network learning process must be stated.

As we remember the perfect map for the given joint probability distribution $\mathcal{P}$ of the set of variables $\mathcal{V}$ is a DAG $\mathcal{G}$ operating on the vertices set $\mathcal{V}$ satisfying $Ind_{\mathcal{G}} = Ind_{\mathcal{P}}$, that is containing d-separations exactly corresponding to conditional independencies existing in $\mathcal{P}$. As we have seen in Theorem 2.4.2.1 the more d-separations a given graph contains, the sparser it is. A perfect map is as a consequence the sparsest possible graph which can represent in the form of Bayesian network a desired distribution, as a given DAG can represent only these joint probability distributions which satisfy all its d-separations in the form of conditional independencies.

In the case when there is no guarantee that the ideal network structure—the perfect map for the given distribution—exists, we can still search for the structure as close to such ideal as possible. Namely, the following definition reflects what we would like to find.

Definition 3.2.1.1 (inclusion optimal map) *Let $\mathcal{P}$ be the joint probability distribution of some given set of random variables $\mathcal{V}$. We say that a DAG $\mathcal{G}$ operating on the set of vertices $\mathcal{V}$ is the inclusion optimal map for the distribution $\mathcal{P}$ if $Ind_{\mathcal{G}} \subseteq Ind_{\mathcal{P}}$ (equivalently: the pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network) and there does not exist any other DAG $\mathcal{H}$ operating on $\mathcal{V}$ such that $Ind_{\mathcal{G}} \subset Ind_{\mathcal{H}} \subseteq Ind_{\mathcal{P}}$.*

The inclusion optimal map for the given distribution is not unique. In particular if some graph is an inclusion optimal map, then any other DAG within its Markov equivalence class is also an inclusion optimal map—as they all contain exactly the same d-separations. So instead of searching for one particular inclusion optimal map we can consider this search on the higher level of abstraction and look for the Markov equivalence classes of inclusion optimal maps. Whenever the generative distribution admits a faithful representation there is only one Markov equivalence class of inclusion optimal maps—the class of perfect maps for this distribution.

3.2.2. Dimension

When the distribution does not admit a faithful representation there might be potentially many classes of inclusion optimal maps. So the question is which we would like to find most of all. The following definition reflects an additional characterization of the Bayesian network structure reflecting some aspect of the sparsity of this model. This characterization is a fundamental criterion on the basis of which some classes of inclusion optimal maps are preferred over others in the score-based learning branch of algorithms.

Definition 3.2.2.1 (dimension) *The dimension of a given DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is some set of random variables is the number of the parameters required to express the conditional distribution of each vertex—variable given its parents in $\mathcal{G}$. In other words, the dimension is*

equal to:

$$\sum_{X \in \mathcal{V}} (|Dom_X| - 1) \prod_{Y \in \pi_X^{\mathcal{G}}} |Dom_Y|.$$

The above equation can be explained as follows. For each variable X we need to store the conditional distribution of X given any possible configuration of values of its parents. But for each such configuration the conditional distribution of X can be expressed by $|Dom_X - 1|$ parameters—as the probability of the last possible value of X can be inferred from the probabilities of all its remaining possible values.

For example, for the case of the DAG presented in Figure 3.2.2.1, operating on four nodes—random variables A, B, C and D, where A, B and C are binary while D has three possible values, there are 2 parameters describing the conditional distribution of A, 4 corresponding to B, 1 corresponding to C and 4 corresponding to D—so totally we obtain that the dimension of this network is 11.

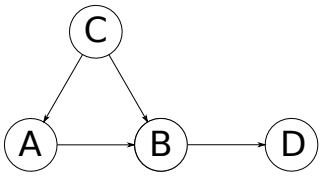


Figure 3.2.2.1: Picture of an exemplary DAG, on the basis of which the dimension concept is exemplified.

The dimension concept is indeed a reasonable sparsity characterization. It is significantly correlated with the so far introduced property which according to Theorem 2.4.2.1 is reflecting the sparsity of the model very well—namely with the d-separation notion. First of all, there holds the following, very simple to prove, property. The proof can be found for example in the Neapolitan (2003) book.

Remark 3.2.2.1 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{H} = (\mathcal{V}, \mathcal{E}_2)$ operating on the same set of nodes—random variables $\mathcal{V}$, where $\mathcal{H}$ is the result of reversing some covered edge in $\mathcal{G}$. Then the dimension of both DAGs $\mathcal{G}$ and $\mathcal{H}$ is the same.*

Let us recall that according to Theorem 2.4.2.1 the condition that for some two DAGs operating on the same set of vertices $\mathcal{G}$ and $\mathcal{H}$ there holds $Ind_{\mathcal{G}} \subseteq Ind_{\mathcal{H}}$ is equivalent to the condition that $\mathcal{H}$ can be obtained from $\mathcal{G}$ after a sequence of operations corresponding to edges deletion and reversion of currently covered edges. So in particular two DAGs have the same d-separations if and only if there is a sequence of currently covered edges reversion leading from the one graph to the second. The reason is simple—any edge deletion in the aforementioned sequence of operations permanently leads us to some other Markov equivalence class (during further operations corresponding only to reversing and removing some edges the skeleton of the transformed graph would not grow again—so the skeleton would be permanently changed, which according to Theorem 2.4.3.1 means that the set of d-separations of the graph after the sequence of operations would be different).

So in particular the graphs $\mathcal{G}$ and $\mathcal{H}$ from Remark 3.2.2.1 are Markov equivalent, and more generally two DAGs are Markov equivalent if and only if there is a sequence of the covered edges reversions leading from the one graph to the second. In conjunction with Remark 3.2.2.1 we obtain the following conclusion.

Remark 3.2.2.2 *In any Markov equivalence class all belonging to it graphs have the same dimension.*

In other words two DAGs operating on the same set of vertices and having the same d-separations have the same dimension.

3.2.3. Parameter Optimal Map

But the connection between the dimension concept and d-separation concept is even stronger. Notice that any edge removal in a DAG always leads to the reduction of its dimension—simply because the only changes in parameters describing conditional distributions of each node given its parents in the graph are related to the vertex to which the removed edge was leading. It has one parent less—so the number of possible configurations of values of its parents is decreased (we assume that each variable, in particular this removed parent, has at least two possible values), and thus the number of parameters describing its conditional distribution is decreased too. If we connect this observation with Theorem 2.4.2.1 and Remark 3.2.2.1 then we obtain the next instant conclusion

Remark 3.2.3.1 *If DAGs $\mathcal{G}$ and $\mathcal{H}$ operating on the same set of vertices—random variables satisfy $Ind_{\mathcal{G}} \subset Ind_{\mathcal{H}}$ then the dimension of $\mathcal{H}$ is lower than the dimension of $\mathcal{G}$.*

It is strictly lower because in the sequence of operations described in Theorem 2.4.2.1 leading from DAG $\mathcal{G}$ to $\mathcal{H}$ there must be at least one edge removal—otherwise according to our previous consideration we would have $Ind_{\mathcal{G}} = Ind_{\mathcal{H}}$.

What the above remarks are intended to lead us is the definition of an alternative to the perfect map notion of the Bayesian network structure optimality—stronger than the inclusion optimal map and similarly as it existing always—in particular also when the considered joint probability distribution does not admit a faithful representation. As we will see soon the theoretical aim of the score-based Bayesian network structure learning algorithms is exactly the defined below structure.

Definition 3.2.3.1 (parameter optimal map) *Let $\mathcal{P}$ be the joint probability distribution of some given set of random variables $\mathcal{V}$. We say that a DAG $\mathcal{G}$ operating on the set of vertices $\mathcal{V}$ is the parameter optimal map for the distribution $\mathcal{P}$ if $Ind_{\mathcal{G}} \subseteq Ind_{\mathcal{P}}$ (equivalently: the pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network) and there does not exist any other DAG $\mathcal{H}$ operating on $\mathcal{V}$ such that $Ind_{\mathcal{H}} \subseteq Ind_{\mathcal{P}}$ and $\mathcal{H}$ has lower dimension than $\mathcal{G}$.*

A parameter optimal map has several important properties. First of all, similarly as it was in the case of inclusion optimal maps, if some DAG is a parameter optimal map then any other DAG in its Markov equivalence class is a parameter optimal map too—as according to Remark 3.2.2.2 all DAGs within one Markov equivalence class have the same dimension. So instead of looking at particular graphs we can think in terms of these classes and try to find some class of parameter optimal maps.

Secondly, a parameter optimal map is indeed a stronger notion than an inclusion optimal map, in the sense that every parameter optimal map is an inclusion optimal map. The reason is following. If a DAG $\mathcal{G}$ is the parameter optimal map for the distribution $\mathcal{P}$, then any other DAG $\mathcal{H}$ satisfying $Ind_{\mathcal{G}} \subset Ind_{\mathcal{H}} \subseteq Ind_{\mathcal{P}}$ would according to Remark 3.2.3.1 have smaller dimension than $\mathcal{G}$—which stays in contradiction with the fact that $\mathcal{G}$ is a parameter optimal map.

Finally, it must be pointed out that the parameter optimal map is a direct generalization of the perfect map concept—similarly as the inclusion optimal map. Namely, whenever

the considered joint probability distribution admits a faithful representation there is only one Markov equivalence class of inclusion optimal maps and one class of parameter optimal maps—and both these classes correspond to exactly the same one Markov equivalence class of perfect maps.

3.2.4. Scoring Criteria

In the world of constraint-based Bayesian network structure learning methods the main indicators are performed statistical tests for conditional independence. In the world of score-based learning methods there is an another indicator, called scoring criterion. The scoring criterion $Score$ is a two-arguments function. It assigns a real value to any pair $(\mathcal{D}, \mathcal{G})$, where $\mathcal{D}$ is a dataset and $\mathcal{G}$ is a DAG consisting of nodes corresponding to attributes of the dataset $\mathcal{D}$. The intuitive understanding of any such score is that it provides us an information how adequate is $\mathcal{G}$ in the role of the structure of a Bayesian network representing the generative distribution of the given sample $\mathcal{D}$. The higher the score, the better is the model in the context of representing this distribution.

There exists many scoring criteria, each having some specific properties. We will not be interested in this work about any further details. Let us only mention that typically the origins of such criteria are appropriate theoretical probabilistic frameworks, in which the value $Score(\mathcal{D}, \mathcal{G})$ corresponds to the probability of the generation of the sample exactly the same as the dataset $\mathcal{D}$ given that the the generative distribution of this sample is represented with regard to the considered framework by a Bayesian network of the structure $\mathcal{G}$. A reader interested in the details of constructions of several classical scoring criteria, as well as some of their properties, can for example study appropriate chapters from the Neapolitan (2003) book.

3.2.5. Score Consistency

Whichever scoring criterion we use, there is one especially critical and desirable property of such criterion—in fact the fundamental property in the domain of score-based learning algorithms—called consistency.

Definition 3.2.5.1 (consistency) *We say that the scoring criterion Score is consistent if for every joint probability distribution $\mathcal{P}$ of any set of random variables $\mathcal{V}$ the following property holds: for any DAGs $\mathcal{G}$ and $\mathcal{H}$ operating on the set of vertices $\mathcal{V}$ such that either*

1. *the pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network and the pair $(\mathcal{H}, \mathcal{P})$ is not, or*
2. *both $(\mathcal{G}, \mathcal{P})$ and $(\mathcal{H}, \mathcal{P})$ are Bayesian networks but $\mathcal{G}$ has smaller dimension than $\mathcal{H}$,*

the probability, with respect to $\mathcal{D}_m$—the sample of size m generated from the distribution $\mathcal{P}$, of the inequality

$$Score(\mathcal{D}_m, \mathcal{G}) > Score(\mathcal{D}_m, \mathcal{H}) \tag{3.2.5.1}$$

goes to 1 when m goes to infinity.

Let us analyze what is the consequence of the limit property defining the consistency notion. Let us think about some particular joint probability distribution $\mathcal{P}$ describing some set of random variables $\mathcal{V}$, and let us consider the family of all possible DAGs operating on the set of vertices $\mathcal{V}$. This family is finite—as we consider only the finite sets of random variables in all this dissertation. So as a consequence a consistent scoring criterion $Score$ satisfies that

the probability that for every two DAGs $\mathcal{G}$ and $\mathcal{H}$ from this family satisfying at least one of the conditions 1 or 2 from the definition there holds inequality 3.2.5.1 goes to 1 when m goes to infinity.

Let us notice that if for every two DAGs $\mathcal{G}$ and $\mathcal{H}$ operating the set of vertices $\mathcal{V}$ and satisfying at least one of the conditions 1 or 2 from the definition the inequality $Score(\mathcal{D}, \mathcal{G}) > Score(\mathcal{D}, \mathcal{H})$ with respect to some scoring criterion $Score$ and dataset $\mathcal{D}$ holds, then the function $Score(\mathcal{D}, .)$ clearly takes a maximum for some parameter optimal map of $\mathcal{P}$, where $\mathcal{P}$ is the joint probability distribution of the set of variables $\mathcal{V}$.

The immediate conclusion is that for the consistent scoring criterion $Score$ the probability that the value $Score(\mathcal{D}_m, .)$ takes a maximum for some parameter optimal map of the common generative distribution of the generated samples $\mathcal{D}_m$ goes to 1 when m goes to infinity. In other words, asymptotically with the grow of the sample size the process of searching for the network structure maximizing some consistent scoring criterion directly leads us to finding the optimal structure—a parameter optimal map.

3.2.6. Score-Based Learning

The score-based learning methods are simply focused on trying to find as highly scored DAG for the available sample as possible—or alternatively a Markov equivalence class of highly scored DAGs, in the case of the scores evaluating equally equivalent structures. The problem, as we will see in the moment, is not easy, thus in practical applications most often different heuristic strategies are applied by researchers in order to perform this search. One of the fundamental score-based learning algorithms, the root of all this branch of methods, like SGS in the case of constraint-based learning algorithms, is a simple greedy search algorithm called K2 (Cooper and Herskovits, 1992). The algorithm has been proposed in this article together with one of the fundamental nowadays scoring criteria—the Bayesian scoring criterion, sometimes called a BD metric (Bayesian-Dirichlet metric). In the typical setting (this score is equipped with the user-defined parameters), where no structures are ruled-out a-priori, it is a consistent criterion.

The development in the area of the score-based learning field has focused on several aspects, like:

- Enhancing the already existing scoring criteria, and designing new. For example, the important modification of the Bayesian scoring criterion has been proposed by Heckerman et al. (1995). The proposed here criterion, so called BDe metric, which is in fact a special case of the BD metric resulting from assigning appropriately the user parameters of this criterion, in particular assures that the Markov equivalent network structures receive the same score.

- Designing new searching procedures, very often heuristic and aimed at finding a particular DAG, but sometimes also exact as well as operating on the Markov equivalence classes. An example of an exact and operating on the classes algorithm is a very notable Greedy Equivalent Search method (Chickering, 2002), to which we will return in Section 7, as it was one of the compared algorithms in the experiments reported in this chapter.

- Research in the theoretical fields reflecting the hardness of Bayesian network structure learning—expressed in terms of the score-based paradigm.

In the remaining part of this section we will focus on some most significant results related to the last point.

3.2.7. Hardness of Finite Sample Bayesian Network Structure Learning

Similarly as in the case of constraint-based algorithms we in practice are not having access to a sample of infinite size. Only some finite portion of data is available, and as the result also in the score-based learning branch we cannot assure finding the perfect solution on the basis of such sample. But in contrary to the constraint-based learning domain even in the case of a finite and unreliable sample there is a strictly defined aim—finding the DAG maximizing a scoring criterion for the available data. Such DAG maximizing the criterion is not necessarily (and in practical applications of high dimensional datasets definitely not) the desired optimal structure representing the generative distribution of the available sample—only asymptotically with the grow of the sample we can be sure that we find such solution. However the strictly defined optimization problem is in this branch of algorithms clearly formulated. This has enabled the research focusing on the hardness of the Bayesian network structure learning problem with regard to such optimization formulation.

The important step towards understanding the hardness of the Bayesian network structure learning has been achieved by Chickering (1996). For any fixed positive integer k he has considered the following decision problem.

Bayesian-Network-Structure-Learning(k)

Given on the entry a dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$, a scoring criterion $Score$ and a real value p, determine whether there exists a DAG $\mathcal{G}$ operating on the set of nodes $\mathcal{V}$ where each node has at most k parents and such that $Score(\mathcal{D}, \mathcal{G}) > p$.

Chickering has noticed first of all that some previous complexity results in the related topic achieved by Höffgen (1993) can be easily transformed into the proof of the NP-hardness of the Bayesian-Network-Structure-Learning(k) problem for $k > 1$ and the input criterion $Score$ limited to the form of the BD metric. Moreover, he has proved that even a more limited version of this decision problem where $k > 1$ and the input criterion $Score$ is the BDe metric is still NP-hard. The hardness of the mentioned decision problems can be automatically translated to the hardness of the corresponding optimization problems where we search for the highest scored according to the criterion $Score$ (BD and especially BDe) DAG where each vertex has at most k parents, for any fixed $k > 1$.

3.2.8. Hardness of Asymptotic Bayesian Network Structure Learning

Although the above results have shown that the finite-sample learning is NP-hard when we use one of the most frequently applied Bayesian scoring criterion, there was still some hope that at least in some aspects the hardness of the learning is lower. In particular the complexity results given by Chickering (1996) does not embrace any other than the BD or BDe metric consistent scoring criterion. Moreover, the consistent criteria according to our previous considerations become regular asymptotically with the grow of the sample size—as they favor in such case the perfect model for the generative distribution—a parameter optimal map. So there might arise an interesting question: what happens in the situation when on the entry we have access to the generative distribution itself ? Such situation would directly correspond to the purely theoretical setting where on the entry we have access to a sample perfectly reflecting the generative distribution, which in particular means that it must be sufficiently large.

Of course whatever is the length of the sample there is no any guarantee that it perfectly reflects the generative distribution—simply because it is just a randomly generated dataset. But with the grow of the sample we can expect to observe the dataset very well reflecting

this distribution, and in particular the probability that the consistent scoring criterion leads us to the perfect solution goes in such case to 1. Does the regularity of the measures in such asymptotic scenario leads to the reduction of the pessimistic searching complexity ? In particular, is it easy to find a parameter optimal map for the given joint probability distribution, assuming that we have a direct access to it ?

The above formulated doubts and questions have been definitely resolved by Chickering et al. (2004). The authors have formulated the following decision problem strictly corresponding to the mentioned asymptotic scenario.

Asymptotic-Bayesian-Network-Structure-Learning

Given on the entry the joint probability distribution $\mathcal{P}$ of some set of random variables $\mathcal{V}$ and a positive integer d, determine whether there exists a DAG $\mathcal{G}$ operating on the set of nodes $\mathcal{V}$ such that the pair $(\mathcal{G}, \mathcal{P})$ is a Bayesian network and the dimension of $\mathcal{G}$ does not exceed d.

The joint probability distribution on the entry is represented as an assignment of the probability to each possible configuration of values of the set of variables $\mathcal{V}$.

Chickering et al. (2004) have shown that the problem Asymptotic-Bayesian-Network-Structure-Learning is NP-hard. Moreover, they have proved the NP-hardness of this problem subject to several simplifying assumptions. For example the problem has turned out to be NP-hard even when there is available an access to any of the following oracles:

- an independence oracle: in a constant time we can determine whether there holds $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ for any $X, Y \in \mathcal{V}$ and $\mathbb{Z} \subseteq \mathcal{V}$,
- a constrained inference oracle: in a constant time we can compute $\mathcal{P}(\mathbb{Z} = \mathbb{z})$ for any $\mathbb{Z} \subseteq \mathcal{V}$ and $\mathbb{z} \in Dom_{\mathbb{Z}}$ such that $|\mathbb{Z}| \leqslant k$ for some constant k,
- a constrained information oracle: in a constant time we can compute the conditional mutual information with regard to the $\mathcal{P}$ distribution $MI_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ for any $X, Y \in \mathcal{V}$ and $\mathbb{Z} \subseteq \mathcal{V}$ such that $|\mathbb{Z}| \leqslant k$ for some constant k.

They have even proved for any integer $k > 2$ the NP-hardness of this problem when we allow the above mentioned access to the oracles and when the search is restricted to DAGs in which each node has at most k parents.

Similarly as in the case of the previously mentioned Chickering (1996) results the considered here NP-hardness of the several versions of decision problems can be immediately transformed into the NP-hardness of the corresponding optimization problems, where we search for the parameter optimal map, or more generally for the graph of the lowest dimension among those representing the desired distribution in the case of appropriately limited search space).

Both the aforementioned theoretical results corresponding to the NP-hardness of the structure learning in the finite-sample and asymptotic case are considered nowadays as the fundamental results in the computational complexity theory related to the Bayesian network structure learning. Especially notable is the result in the asymptotic case. According to it searching for an optimal structure with an assist of any consistent scoring criterion is hard even in the most favorable for these measures case, where the asymptotic regularity of these measures is assured.

3.3. Novel View on Hardness of Bayesian Network Structure Learning

We have reached the point where the first, purely theoretical result of this dissertation is presented. We show in this section an alternative view on the NP-hardness of the Bayesian network structure learning problem. The origin of this section is the Betliński and Ślęzak (2012) article.

We first state some of the motivations standing behind the consideration of the hardness of the introduced here scenario. Then we state the problem together with some illustrative example, and we propose and prove the correctness of the appropriate polynomial transformation, the consequence of which is the NP-hardness of this problem. Finally we sketch some possible directions of the future work by pointing out the potential improvements of the result.

3.3.1. Motivations

Let us first tell something about the motivations of the presented here research. Some potential problem with the classical view on the NP-hardness of the network structure learning provided by Chickering (1996) is that it might be potentially considered as unintuitive—especially for a reader not familiar with the Bayesian networks domain. The problem of finding the optimal structure is seen here as the problem of maximizing some scoring criterion. Any standard criterion is itself inherited from quite advanced statistical considerations, for example the Bayesian scoring criterion is derived from the world of multidimensional Dirichlet distributions. In order to understand well the genesis of such measure a relatively advanced orientation of the reader in the statistics domain is required.

From the other side, even if the understanding of the genesis of such measure is not a problem for the reader, then the question might still remain: what the searching for the optimal according to the given scoring criterion structure in the case of the available finite sample actually means when we consider it in isolation from the whole theoretical framework from which it is derived ? What we know for sure is that asymptotically with the grow of the sample such search leads to a parameter optimal map of the generative distribution. But in the case of a finite sample the DAG achieving the highest score does not have any obvious interpretation.

Theoretical results for the asymptotic case provided by Chickering et al. (2004) give a significant insight into the hardness of searching for optimal structures with an assist of consistent scoring criteria. There is a very clear interpretation of the goal here—as we are interested in finding the parameter optimal map. But these results are in fact on a high level of abstraction—as in practice we deal usually with an incomplete knowledge about the generative distribution, in the form of some finite sample—instead of the complete information, as assumed here.

The proposed in this section formulation of the Bayesian network structure learning problem is an attempt to take advantage from both the previously described problem statements. Namely, similarly as in the Chickering (1996) scenario, and in contrast to the Chickering et al. (2004) scenario, we will stick to the completely practical scenario, where on the input we have some finite sample. From the other side, similarly as in the Chickering et al. (2004) scenario, and in contrast to the Chickering (1996) scenario, we will focus on searching for the model which has an easy to understand and interpret nature.

The origin of all the investigations described further is placed in the Ślęzak (2002) work, focused on the topic of approximate decision reducts—strongly related to the aforementioned

in Subsection 2.2.9 Markov blankets. In particular we have adopted some of the techniques appearing in the proofs of the NP-hardness of the stated in this work problems in order to show the NP-hardness of the proposed here problem.

3.3.2. Problem Formulation

The main point of interest of the remaining part of this chapter is the following optimization problem.

Minimal-Bayesian-Network-Structure-Learning

Given on the entry a dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$, find a DAG $\mathcal{G}$ operating on the set of nodes $\mathcal{V}$ such that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{D}})$ is a Bayesian network and having the minimal number of edges.

As we can see the problem is here focused on searching for the DAG representing in the form of a Bayesian network the distribution $\mathcal{P}_{\mathcal{D}}$—that is exactly the one determined by the available sample $\mathcal{D}$. As usually we are interested in finding as sparse Bayesian network structure as possible. Here this sparsity is expressed in a very straightforward way—we simply want to find the DAG representing the distribution of the sample consisting of the minimal number of edges.

We are sticking to the distribution represented explicitly by the input dataset, not to the approximation of the unknown generative distribution, as it happens in the case of the Chickering (1996) formulation. However, in the asymptotic case with regard to the sample size the probability of the ability to retrieve the perfect network structure from distribution $\mathcal{P}_{\mathcal{D}}$ goes to 1—thus asymptotically both the problem formulations lead to the similar goal.

The sparseness condition is here also different than in the case of the theoretical aim of score-based methods. We are searching here for the Bayesian network structure consisting of the minimal number of edges, while in the classical view the aim is the Bayesian network structure having the minimal number of parameters. These two conditions are not equivalent in general.

In the special case, where the generative distribution admits a faithful representation, there exists one Markov equivalence class of parameter optimal maps—corresponding to the class of perfect maps for this distribution. The perfect map has in particular the minimal number of edges among all DAGs which can represent the desired distribution. Any other than a perfect map DAG having the same skeleton definitely does not represent this distribution—as there is no sequence of edges removals and covered edges reversions leading from this DAG to such perfect map (there could be only edge reversals in this sequence in fact as the skeletons are the same—but such operations would not lead to another Markov equivalence class). So any DAG representing the generative distribution having the minimal number of edges is a perfect map, whenever this distribution admits a faithful representation.

This means that in the case when the perfect map for the generative distribution exists, asymptotically with the grow of the sample the search aim defined in the problem Minimal-Bayesian-Network-Structure-Learning and the typical search aim of finding a DAG maximizing some consistent scoring criterion are leading to the same desired structure—the perfect map for the generative distribution.

3.3.3. Applied Polynomial Reduction

We will now prove the NP-hardness of the Minimal-Bayesian-Network-Structure-Learning problem. The proof is by the polynomial reduction of the following well-known NP-hard problem to Minimal-Bayesian-Network-Structure-Learning:

MINIMUM-DOMINATING-SET

Given on the entry an undirected graph, find the minimal dominating set for this graph, that is such minimal subset of vertices, that every vertex of the graph is either in this subset, or it is a neighbor of some element of this subset.

Assume that on the entry of an instance of the MINIMUM-DOMINATING-SET problem we have an undirected graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{X_1, ..., X_n\}$. The proposed reduction $\mathcal{T}$ transforms it into the instance of the MINIMAL-BAYESIAN-NETWORK-STRUCTURE-LEARNING problem, where on the entry we have a dataset $\mathcal{D}_\mathcal{H} = (\mathcal{U}_\mathcal{H}, \mathcal{V}_\mathcal{H}), \mathcal{U}_\mathcal{H} = \{U_1, ..., U_{2n+1}\}$, $\mathcal{V}_\mathcal{H} = \{X_1, ..., X_n, T\}$, where:

- for $1 \leqslant i, j \leqslant n$ $\mathcal{D}_\mathcal{H}[U_i, X_j] = 1$ if $i = j$ or $(X_i, X_j) \in \mathcal{E}$, otherwise $\mathcal{D}_\mathcal{H}[U_i, X_j] = 0$,
- $\mathcal{D}_\mathcal{H}[U_{n+1}, X_i] = \mathcal{D}_\mathcal{H}[U_i, T] = 0$ for $1 \leqslant i \leqslant n$,
- $\mathcal{D}_\mathcal{H}[U_{n+1}, T] = 1$,
- $\mathcal{D}_\mathcal{H}[U_{n+1+i}, X_i] = 3$ for $1 \leqslant i \leqslant n$,
- all remaining cells in $\mathcal{D}_\mathcal{H}$ have value 2.

For example, if on the entry of an instance of the MINIMUM-DOMINATING-SET problem we have a graph $\mathcal{H}$ presented in Figure 3.3.3.1, then the above described reduction $\mathcal{T}$ transforms it into the instance of the MINIMAL-BAYESIAN-NETWORK-STRUCTURE-LEARNING problem, where on the entry there is a dataset $\mathcal{D}_\mathcal{H}$ presented in Figure 3.3.3.2.

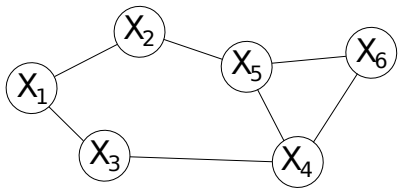


Figure 3.3.3.1: An undirected graph $\mathcal{H}$.

$\mathcal{D}_\mathcal{H}$	X_1	X_2	X_3	X_4	X_5	X_6	T
U_1	1	1	1	0	0	0	0
U_2	1	1	0	0	1	0	0
U_3	1	0	1	1	0	0	0
U_4	0	0	1	1	1	1	0
U_5	0	1	0	1	1	1	0
U_6	0	0	0	1	1	1	0
U_7	0	0	0	0	0	0	1
U_8	3	2	2	2	2	2	2
U_9	2	3	2	2	2	2	2
U_{10}	2	2	3	2	2	2	2
U_{11}	2	2	2	3	2	2	2
U_{12}	2	2	2	2	3	2	2
U_{13}	2	2	2	2	2	3	2

Figure 3.3.3.2: A dataset $\mathcal{D}_\mathcal{H}$.

We will further very often refer to the joint probability distribution of attributes induced from $\mathcal{D}_\mathcal{H}$, that is to the distribution $\mathcal{P}_{\mathcal{D}_\mathcal{H}}$—as our optimization problem of finding the minimal Bayesian network focuses on representing with an assist of this network exactly this distribution. So for the sake of convenience let us simplify this notation, and denote the distribution $\mathcal{P}_{\mathcal{D}_\mathcal{H}}$ as $\mathcal{P}_\mathcal{H}$.

3.3.4. Proof of Correctness

The remaining part of this section provides four lemmas together with their proofs. Their aim is to show step by step that the proposed transformation $\mathcal{T}$ is indeed a polynomial reduction from the MINIMUM-DOMINATING-SET problem to the MINIMAL-BAYESIAN-NETWORK-STRUCTURE-LEARNING problem, and as a consequence the latter problem is NP-hard.

We will use further all the notations introduced during the description of the $\mathcal{T}$ reduction. In particular we will consider the setting where $\mathcal{T}$ transforms some undirected graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ into the dataset $\mathcal{D}_{\mathcal{H}} = (\mathcal{U}_{\mathcal{H}}, \mathcal{V}_{\mathcal{H}})$ determining the joint probability distribution $\mathcal{P}_{\mathcal{H}}$, where $\mathcal{U}_{\mathcal{H}} = \{U_1, ..., U_{2n+1}\}$ and $\mathcal{V}_{\mathcal{H}} = \{X_1, ..., X_n, T\}$ are defined as previously.

Lemma 3.3.4.1 *Let $\mathcal{G}$ be an arbitrary DAG operating on the set of nodes $\mathcal{V}_{\mathcal{H}}$ such that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ is a Bayesian network. Then for every $1 \leqslant i < j \leqslant n$ vertices X_i and X_j are connected by an edge in $\mathcal{G}$.*

Proof Assume that some $1 \leqslant i < j \leqslant n$ vertices X_i and X_j are not connected by an edge in $\mathcal{G}$. Then according to Remark 2.4.4.1 X_i and X_j are d-separated by some subset $\mathbb{Z}$ of the remaining vertices in $\mathcal{G}$. According to Corollary 2.3.4.1 this means that in the distribution$\mathcal{P}_{\mathcal{H}}$ variables X_i and X_j are independent given $\mathbb{Z}$. This leads to the contradiction—thanks to n last rows of $\mathcal{D}_{\mathcal{H}}$ variables X_i and X_j are dependent given any subset of remaining variables. Namely, given that all variables from a subset are equal to 2 the following conditional dependence of X_i and X_j is observed:

- $X_j = 2$ with probability 1 given that $X_i = 3$,
- $X_j = 3$ with nonzero probability given that $X_i = 2$.

■

Lemma 3.3.4.2 *For an arbitrary subset of nodes $\mathbb{Z} \subseteq \mathcal{V}$ it holds that $\mathbb{Z}$ is a dominating set in $\mathcal{H} \iff$ the conditional independence $Ind_{\mathcal{P}_{\mathcal{H}}}(T, (\mathcal{V}_{\mathcal{H}} \setminus (\{T\} \cup \mathbb{Z})) \mid \mathbb{Z})$ holds.*

Proof $\implies$ We will show that for any configuration of values of the set of attributes $\mathbb{Z}$ appearing in $\mathcal{D}_{\mathcal{H}}$ the number of different occurring in the dataset values of T is reduced to 1. Let us check all cases:

- When all attributes from $\mathbb{Z}$ are equal to 2 or 3, then T must be equal to 2.
- When all attributes from $\mathbb{Z}$ are equal to 0 or 1 and at least one attribute from $\mathbb{Z}$ is equal to 1, then T must be equal to 0.
- When all attributes from $\mathbb{Z}$ are equal to 0, then, thanks to the assumption that $\mathbb{Z}$ is a dominating set in $\mathcal{H}$, the only possible object for which the condition can hold is U_{n+1}, so T must be equal to 1.

$\impliedby$ Assume that $\mathbb{Z}$ is not a dominating set in $\mathcal{H}$. Then there is at least one vertex X_l in $\mathcal{H}$, which is neither included in $\mathbb{Z}$, nor connected by an edge in $\mathcal{H}$ with any element of this subset. So in $\mathcal{D}_{\mathcal{H}}$ the object U_l has value 0 on attributes from $\mathbb{Z}$. An another object with the same pattern on $\mathbb{Z}$ is U_{n+1}. Therefore, there is some nonzero probability with regard to the distribution $\mathcal{P}_{\mathcal{H}}$ of both events $\{T = 0\}$ and $\{T = 1\}$ subject to the zeros setting on $\mathbb{Z}$. Now, let us consider the additional influence on T of variable X_l given the zeros setting on $\mathbb{Z}$. We can observe that when additionally $X_l = 1$ (there is at least one row satisfying this—U_l), then T is equal to 0 with probability 1. Hence, in particular X_l and T are not independent with regard to $\mathcal{P}_{\mathcal{D}}$ given $\mathbb{Z}$, which leads us to the contradiction. ■

Before proceeding further let us make in this moment one comment. The reduction $\mathcal{T}$ which we are currently investigating has turned out after the last lemma to be useful in

order to prove some sideway fact. Notice, that we have already shown that $\mathcal{T}$ is a polynomial reduction from the Minimum-Dominating-Set problem to the following problem:

Markov-Blanket-Learning

Given on the entry a dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ and some variable $T \in \mathcal{V}$ find a Markov blanket of T with regard to the distribution $\mathcal{P}_{\mathcal{D}}$.

Let us remind that according to Definition 2.2.9.1 the Markov blanket of some given variable T is a minimal subset of remaining variables $\mathbb{Z}$ satisfying the conditional independence property formulated in the Lemma 3.3.4.2. According to this lemma the reduction $\mathcal{T}$ transforms any graph $\mathcal{H}$ into such dataset $\mathcal{D}_{\mathcal{H}}$ that all dominating sets in $\mathcal{H}$ correspond to subsets of variables satisfying the mentioned conditional independence—and vice versa. So the Markov blanket of T in the dataset $\mathcal{D}_{\mathcal{H}}$ corresponds to the minimal dominating set in the graph $\mathcal{H}$.

As the result we have obtained that the Markov-Blanket-Learning problem is NP-hard. But let us return to the main scope and continue the analysis of the reduction $\mathcal{T}$ in the main here context.

Lemma 3.3.4.3 *Let $\mathbb{Z}$ be an arbitrary dominating set in $\mathcal{H}$. Then there exists a DAG $\mathcal{G}$ operating on the set of nodes $\mathcal{V}_{\mathcal{H}}$ such that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ is a Bayesian network, in which the set of neighbors of T is $\mathbb{Z}$.*

Proof Let us directly provide the construction of the desired DAG $\mathcal{G}$.

Starting from an empty graph operating on the set of nodes $\mathcal{V}_{\mathcal{H}}$, from every vertex in the set $\mathbb{Z}$ we draw an edge to every vertex not belonging to this set. All not connected yet pairs of vertices from the set $\{X_1, ..., X_n\}$ we connect with an edge in such way, that the whole graph remains a DAG.

What remains is to prove that such construction of the graph $\mathcal{G}$ is correct—that is the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ is a Bayesian network. We will equivalently show that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ satisfies a Markov condition.

Thanks to Lemma 3.3.4.2 we know that the set $\pi_T^{\mathcal{G}} = \mathbb{Z}$ satisfies that given $\mathbb{Z}$ the variable T is independent from all the remaining variables in $\mathcal{P}_{\mathcal{H}}$, so in particular the Markov condition with respect to the variable T is satisfied.

For variables from the set $\mathbb{Z}$ the Markov condition is satisfied in a trivial way, because for each variable from this set the set of its nondescendants excluding parents is empty (these variables are connected with all other in $\mathcal{G}$).

For each remaining variable $X_l \in \{X_1, ..., X_n\} \setminus \mathbb{Z}$ we have that there is only one nondescendant of X_l which is not a parent of X_l—the vertex T. So in order to check the Markov condition for X_l we need to prove the independence in $\mathcal{P}_{\mathcal{H}}$ of X_l and T given $\pi_{X_l}^{\mathcal{G}}$. We do it as follows. We have already checked the Markov condition with respect to T, and we know that T is independent in $\mathcal{P}_{\mathcal{H}}$ even from all other than $\pi_T^{\mathcal{G}}$ variables given $\pi_T^{\mathcal{G}}$. This in particular means, that X_l and T are independent given $\pi_{X_l}^{\mathcal{G}}$ in $\mathcal{P}_{\mathcal{H}}$—because $\pi_T^{\mathcal{G}} \subseteq \pi_{X_l}^{\mathcal{G}}$. ■

For the sake of convenience we will introduce one more notion. Let us recall that the $D[...]$ notation appearing below we have introduced in Subsection 2.1.1.

Definition 3.3.4.1 (domination in a dataset) *Let $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ be a dataset. Let us consider the subsets $\emptyset \neq \mathbb{U} \subseteq \mathcal{U}$ and $\emptyset \neq \mathbb{V} \subseteq \mathcal{V}$, as well as the attribute $W \in \mathcal{V} \setminus \mathbb{V}$. We say that:*

- *The subset of attributes $\mathbb{V}$ dominates in the dataset $\mathcal{D}$, if for every $U \in \mathcal{U}$ such that $\mathcal{D}[U, X] = 0$ for all $X \in \mathbb{V}$ there holds $\mathcal{D}[U, X] = 0$ for all $X \in \mathcal{V}$.*

- *The subset of attributes $\mathbb{V}$ dominates the attribute W in the dataset $\mathcal{D}$, if $\mathbb{V}$ dominates in $\mathcal{D}[\mathcal{U}, \mathbb{V} \cup \{W\}]$.*

- *The subset of attributes $\mathbb{V}$ dominates the attribute W on the subset of objects $\mathbb{U}$ in the dataset $\mathcal{D}$, if $\mathbb{V}$ dominates in $\mathcal{D}[\mathbb{U}, \mathbb{V} \cup \{W\}]$.*

Lemma 3.3.4.4 *Let $\mathcal{G}$ be an arbitrary DAG operating on the set of nodes $\mathcal{V}_{\mathcal{H}}$ such that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ is a Bayesian network. Let $\mathbb{Z}$ be the set of neighbors of the node T in $\mathcal{G}$. Then the subset $\mathbb{Z}$ is a dominating set in $\mathcal{H}$.*

Proof The graph $\mathcal{G}$ is a DAG, so there exists an ancestral order of its vertices, such that every variable has all its parents before itself in the order. Let $[X_{j_1}, ..., X_{j_t}, T, X_{j_{t+1}}, ..., X_{j_n}]$ be this order. In particular, before the vertex T there are placed all its parents, while after T—all its children. We will first consider the case, when the vertex T has at least one parent and at least one child.

The first observation is that $\pi_T^{\mathcal{G}}$ dominates in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_t}\}]$. The reason is following. The pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ satisfies the Markov condition, so in particular T is conditionally independent from variables $\{X_{j_1}, ..., X_{j_t}\} \setminus \pi_T^{\mathcal{G}}$ given $\pi_T^{\mathcal{G}}$ in the distribution $\mathcal{P}_{\mathcal{H}}$. Assume that $\pi_T^{\mathcal{G}}$ does not dominate in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_t}\}]$. This would mean that there is such object U in $\mathcal{D}_{\mathcal{H}}$ for which all attributes corresponding to $\pi_T^{\mathcal{G}}$ are equal to 0 and some other attribute $X_l \in \{X_{j_1}, ..., X_{j_t}\} \setminus \pi_T^{\mathcal{G}}$ is equal to 1. Given that all parents of T are equal to 0, we have that with nonzero probability T has value 1 in $\mathcal{P}_{\mathcal{H}}$—thanks to the object U_{n+1}. But when we add one more condition $X_l = 1$ (there is at least one row satisfying this—U), then with probability 1 T has value 0. So T is not independent from variables $\{X_{j_1}, ..., X_{j_t}\} \setminus \pi_T^{\mathcal{G}}$ given $\pi_T^{\mathcal{G}}$ in $\mathcal{P}_{\mathcal{H}}$—because T is not independent from X_l given $\pi_T^{\mathcal{G}}$ in $\mathcal{P}_{\mathcal{H}}$, which is a contradiction.

The second observation is that for all $l \in \{j_{t+1}, ..., j_n\}$ we have that if X_l is not a child of T in $\mathcal{G}$, then it is dominated in $\mathcal{D}_{\mathcal{H}}$ by $\pi_{X_l}^{\mathcal{G}}$ on the set of objects $\{U_1, ..., U_n\}$. Namely, from the fact that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ satisfies the Markov condition we obtain that in the $\mathcal{P}_{\mathcal{H}}$ distribution X_l is independent from its nondescendants excluding $\pi_{X_l}^{\mathcal{G}}$ given $\pi_{X_l}^{\mathcal{G}}$—so in particular X_l and T are independent given $\pi_{X_l}^{\mathcal{G}}$. Thus, the same argumentation as for the case of the first observation leads us to the conclusion that the parents of X_l dominate X_l on the set of objects $\{U_1, ..., U_n\}$.

Now we will inductively prove that for each l satisfying $t \leqslant l \leqslant n$, in the dataset $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_l}\}]$ the occurring neighbors of T in $\mathcal{G}$ dominate. For $l = n$ this property means simply that the set $\mathbb{Z}$ dominates in the dataset $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_n}\}] = \mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_1, ..., X_n\}]$—which in turn means that $\mathbb{Z}$ is a dominating set in $\mathcal{H}$.

The first step of the induction has been already proved, as it was the first considered in this proof observation. Now, let us assume that for some l satisfying $t \leqslant l \leqslant n$ in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_l}\}]$ the occurring neighbors of T dominate. If $X_{j_{l+1}}$ is a child of T, then automatically we obtain that in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_{l+1}}\}]$ the occurring neighbors of T dominate. If $X_{j_{l+1}}$ is not a child of T, then, by following the second out of the above observations, $X_{j_{l+1}}$ is dominated by the remaining attributes in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_{l+1}}\}]$, which—thanks to the assumption—are dominated by neighbors of T. Therefore, the occurring neighbors of T dominate in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}, ..., X_{j_{l+1}}\}]$.

What remains to check are special cases:

- When T has only parents in $\mathcal{G}$, then T can be placed at the end of an ancestral order, so the first step of the described above induction ends the proof.

- When T has only children in $\mathcal{G}$, then the only thing necessary to change is the first step of the induction. We do it as follows: Vertex T has no parents, so we can put it at the

beginning in the ancestral order—our order is now $[T, X_{j_1}, ..., X_{j_n}]$. Note that T and X_{j_1} are then connected by an edge (T is the parent) because otherwise, according to the fact that the pair $(\mathcal{G}, \mathcal{P}_{\mathcal{H}})$ satisfies the Markov condition, we would have in particular that in the $\mathcal{P}_{\mathcal{H}}$ distribution X_{j_1} is independent from its nondescendants in $\mathcal{G}$ given its parents, which in our case implies that X_{j_1} and T are independent in $\mathcal{P}_{\mathcal{H}}$—while this cannot be true. In the distribution $\mathcal{P}_{\mathcal{H}}$ the variable T depends on each of the remaining variables—as the event $\{T = 1\}$ has the nonzero probability only in the case when any other variable has value 0. So as the first step in the induction procedure we can consider the fact that in $\mathcal{D}_{\mathcal{H}}[\{U_1, ..., U_n\}, \{X_{j_1}\}]$ the attribute X_{j_1}, that is the occurring here neighbor of T, is in a trivial way dominating in this dataset.

- Let us notice that the argumentation from the point above shows also that it is not possible for T to not have any neighbor in $\mathcal{G}$—as in such case we could place T at the beginning of the ancestral order and again conclude that it must be connected by an edge in $\mathcal{G}$ with the next variable in this order.

■

Now we can easily prove the main result of this section:

Theorem 3.3.4.1 MINIMAL-BAYESIAN-NETWORK-STRUCTURE-LEARNING *is NP-hard.*

Proof From Lemmas 3.3.4.3 and 3.3.4.4 we have, that if we take a DAG of the minimal number of neighbors of T from all DAGs which can represent $\mathcal{P}_{\mathcal{H}}$ in the form of the Bayesian network, then these neighbors correspond to the minimal dominating set in $\mathcal{H}$. Lemma 3.3.4.1 in addition gives us guarantee that a DAG of the minimal number of edges from all which can represent $\mathcal{P}_{\mathcal{H}}$ has also the minimal number of neighbors of T. ■

3.3.5. Discussion

The presented and proved new theoretical result has its advantages, as we have wrote it earlier. We stick here to the practical case where on the entry is some finite sample available, and we use the easy to understand and intuitive optimization criterion.

However, we should point it out very clearly here, that this result is not satisfactory for us. Much of the attention should be taken in the future in order to examine whether it can be extended to some other desirable hypothesis.

First of all, the sparsity criterion which is used in the MINIMAL-BAYESIAN-NETWORK-STRUCTURE-LEARNING problem is not necessarily the one most interesting. It would be in particular very desirable to obtain the analogical NP-hardness result for the same problem with an exception that we search for the DAG representing the sample distribution not of the minimal number of edges, but of the minimal dimension. The dimension of the graph seems to be a more important sparsity indicator than the number of edges. The main problem with the Bayesian networks representation and inference can be caused indeed by the high dimension rather than by the high number of edges. Note that the number of parameters describing the conditional probability of any vertex given its parents in the network structure grows exponentially with the increase of the number of parents. Such grow can cause the proportional increase in the amount of required memory in order to storage the Bayesian network, as well as the drastic decrease in the reliability of such network—simply because the reliability of the conditional distribution representation inferred from a given dataset

rapidly decreases with the grow of the conditional set. Thus the more important than the number of edges can be their direction assuring some reduction of dimension.

On appropriately high level of abstraction the reduction of dimension in the field of Bayesian networks have its correspondence in the field of rough sets. For example, to some extent an analogous problem of searching for decision reducts implying the minimal number of rules was investigated by Ślęzak (2002).

Secondly, although in the Minimal-Bayesian-Network-Structure-Learning problem we try to cover a practical scenario of the Bayesian network structure learning—this coverage could be significantly extended. Imagine that we have some small sample on the input of our optimization problem—such small that it is definitely poorly reflecting the generative distribution. One may ask in such case what is the purpose of learning the sparsest graph strictly representing the available possibly degenerated sample distribution. Clearly the modeling of all the details of this distribution does not make sense. Some another, approximate formulation of the problem would be here rather more welcome. There are several possibilities.

We can for example search for the sparsest DAG which can represent the joint probability distribution sufficiently similar to the distribution strictly describing the sample, for example in the distance to this distribution not exceeding some bound, where some appropriately constructed distance measure is used. This measure can be for example based on appropriately constructed degrees of approximate conditional independence. In particular we might consider in the future employing here the Ślęzak (2009) postulates.

Or we can think about the searching for a DAG representing exactly or approximately the sample distribution but not necessarily a sparsest possible—rather sufficiently sparse, like differing by some factor from the minimal possible number of edges. In other words we might be interested not only in showing the hardness of searching for the optimal solution, but also that the whole problem is also hard to approximate (Papadimitriou, 1994).

Finally, we can even think about mixing the above ideas with the mentioned more desired sparseness criterion based on the dimension notion. The NP-hardness of all these problems remains unresolved. Some of them might be possibly the topic of our future research.

Chapter 4

Constraint-Based Learning—Rudiments

This chapter is the most important part with respect to all remaining and main subjects of this dissertation. We provide here a wide background in the field of the constraint-based Bayesian network structure learning algorithms. We analyze in details three algorithms here: two appearing at the very beginning of this field: SGS and PC, as well as one very significant improvement of them, exploiting the notion of Markov blankets. We in particular return in this chapter to this notion. So far we have introduced it very briefly in Subsection 2.2.9, without showing its real connections with the considered topic. Now we will see that the Markov blanket is in fact a central concept in the Bayesian networks domain, and employing learning of them in Bayesian network structure learning methods had a big impact on the progress in this area.

4.1. Algorithms Analysis Scheme

Before going into the main topic let us first describe according to what aspects we will analyze each of presented further—in this chapter as well as the next one—constraint-based learning algorithm. First we state here several assumptions about the nature and characteristics of the input dataset, with regard to which the whole analysis of each such method is performed. Then we describe with regard to what aspects the performance of any investigated algorithm is measured. Finally we introduce four scenarios of Bayesian network structure learning: two of the theoretical nature and two purely practical. These scenarios correspond to several assumptions about the probabilistic nature of the input dataset, as well as about the nature of conditional independence detection mechanism.

4.1.1. Basic Assumptions

On the entry of each further described algorithm we have a sample $\mathcal{D} = (\mathcal{U}, \mathcal{V})$. Let us remind that the constraint-based algorithms are exploiting statistical tests investigating a conditional independence of the form $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ where $X, Y \in \mathcal{V}$, $X \neq Y$, $\mathbb{Z} \subseteq \mathcal{V} \setminus \{X, Y\}$ and $\mathcal{P}$ is the generative distribution of the sample, assuming it exists. The whole idea standing behind the constraint-based algorithms is that in the case when $\mathcal{P}$ indeed exists and it admits a faithful representation the retrieved knowledge about conditional independencies—assuming it is correct—reflects exactly d-separations occurring in the desired ideal Bayesian network structure, that is a perfect map for the distribution $\mathcal{P}$. According to Remarks 2.4.4.1 and

2.4.4.2 the knowledge about d-separations is sufficient to determine the characterization of the Markov equivalence class of perfect maps: the appropriate skeleton and set of v-structures. Although in practice the theoretical assumptions listed above typically does not hold—the motivation is that even in such a case applying the constraint-based procedure which is correct in the theoretical scenario can lead us to the sensible learned Bayesian network structures.

Any further in this work considered constraint-based algorithm we will analyze with respect to several aspects, where each of them is measured with regard to the size of the input dataset $\mathcal{D}$—namely with regard to the number of rows and columns of this table. We assume here that $\mathcal{D}$ consists of m rows and n columns.

We also as everywhere assume also here that each of the attributes in the input dataset $\mathcal{D}$ corresponds to a random variable having some finite number of possible values—but at least two. Thus we do not take into account any trivial variables, which in the sparse Bayesian network representing the generative distribution would correspond to nodes of degree 0 (there is no dependence of them with respect to all remaining variables). We assume that we use the χ^2 or G^2 test in the role of statistical test of conditional independence.

For the sake of simplifying a bit the time complexity expressions appearing further we also assume that each of the variables in the generative distribution has number of possible values bounded by some common constant v. The time complexity of calculating the χ^2 or G^2 is linear with respect to both m and n (more strictly—with respect to the size of the conditional set), but quadratic with respect to v. Thanks to the assumption of the bounded v we omit the factor v^2 in the time complexity expressions.

4.1.2. Performance Aspects

The analysis of each further introduced algorithm will focus on three aspects: the time complexity of the method, the pessimistic number of performed statistical tests for conditional independence and the pessimistic size of the conditional set processed in these tests, that is the maximal size $|\mathbb{Z}|$ which can appear in the examined $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ relations.

It can be rightly noticed that the first two aspects of the performed analysis are strongly correlated. In fact as we will see the knowledge about the pessimistic number of performed statistical tests for conditional independence and the knowledge about the time complexity of performing each such test very often determines the pessimistic time complexity of the whole Bayesian network learning method. However, in some situations the actual time complexity of the whole approach can be smaller than this pessimistic one, for example when the statistics gathered during performing one test are able to somehow reduce the amount of calculations required in some other test. Whenever such reduction is achievable we will point it out.

The first two aspects are aimed at measuring mostly the time efficiency of the approach. The second aspect however is also correlated with the reliability of the algorithm—less number of performed statistical tests for conditional independence often corresponds to the situation where we are not conducting a lot of these less reliable tests corresponding to the large conditional sets.

The third aspect is directly aimed at measuring the reliability of the Bayesian network learning method. The smaller conditional sets are processed in statistical tests for conditional independence, the less percentage of errors in decisions regarding these relations can be expected, and as a consequence the network structure is constructed on the basis of the more reliable knowledge.

4.1.3. Considered Scenarios

The whole above described analysis we will perform with respect to four scenarios: two having a purely theoretical nature, and two purely practical.

The first theoretical scenario—let us call it ST_1—corresponds to the the purely artificial assumption that we are able to obtain a 100 percent correct information about any conditional independence. Moreover we assume here that the generative distribution of the input sample admits a faithful representation. Let us remind that subject to these assumptions the typical aim of constraint-based algorithms is the perfect map of this generative distribution.

The second theoretical scenario—which we call ST_2—is the same as the first one, but with one additional assumption—in the skeleton characterizing the faithful representation of the generative distribution each vertex has degree at most k (it has at most k neighbors in this graph). The motivation standing behind this scenario is that we want to measure the performance of any Bayesian network learning method in the case when the reference real model is sparse. The sparsity of this model might differently affect compared learning methods, in particular some of them can benefit more on such scenario, while some other less.

The first practical scenario—we call it SP_1—is reflecting exactly the situation which we face in reality. We do not assume here the perfect knowledge about the conditional independencies existing within the attributes set. The statistical tests for conditional independence, on the basis of which any conditional independence relation is examined, are assumed to reflect their real behavior here—in particular some percentage of performed tests leads to wrong decisions. We assume, that we perform these tests without applying the Spirtes et al. (2000) convention of aborting the least reliable tests. We do not assume that there exists the faithful representation of the generative distribution. Moreover, we even do not assume that there exists the generative distribution of the input sample. briefly telling, we are measuring the performance of the considered Bayesian network learning method without any assumptions. This scenario is exactly reflecting the real application of the method.

The second practical scenario—named SP_2—is exactly the same as the first one, with one exception—we apply the Spirtes et al. (2000) convention, namely we assume that the statistical tests for conditional independence examining some relation $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ are not performed when the sample size is smaller than $c|Dom_X||Dom_Y|\prod_{Z\in\mathbb{Z}}|Dom_Z|$, where c is some a priori specified constant. In this case any learning algorithm tries to automatically decide whether to assume the conditional independence or dependence in such unresolved case. As we will further see, depending on the method this automatic decision can be different.

Similarly as the second theoretical scenario, the second practical scenario aims at measuring the performance of the method when some benefits arising from the simplifying assumptions are possible. Limiting the degree of nodes of the faithful representation of the generative distribution would be meaningless here, as we even do not assume in the practical scenarios the existence of such generative distribution. But applying the Spirtes et al. (2000) convention instead might also lead to some benefits in the network learning process, and moreover it is a very common solution used in practice. In particular, as we have mentioned it in Subsection 2.2.8, all the experiments reported in this dissertation were performed with an assist of this solution.

There might arise a question, what is the purpose of considering the theoretical scenario. The answer is that it gives an additional information which might potentially shed light on some differences between the methods in the case when no differences are observable in the practical scenario. Moreover, although the theoretical scenarios reflect an artificial and unreachable in practice setting—to some extend the assumptions occurring there are

observable in practice. As the result the advantage of some method with respect to the theoretical scenario might be a significant indicator of practical usefulness of this approach.

4.2. First Constraint-Based Algorithms: SGS and PC

In this section we introduce two constraint-based Bayesian network structure learning algorithms, which are the root methods in all this field. We in particular explain their theoretical correctness and analyze and compare their performance with regard to all aspects given in Subsection 4.1.2 and all scenarios given in Subsection 4.1.3. We also investigate some practical issues regarding performing statistical tests for conditional independence in these algorithms.

4.2.1. Preface

The SGS algorithm (the name corresponds to the surnames of the authors: Peter Spirtes, Clark Glymour and Richard Scheines) is in fact the root of all the branch of constraint-based Bayesian network structure learning algorithms. It is a very straightforward application of the previously discussed Remarks 2.4.4.1 and 2.4.4.2. The PC algorithm (the name corresponds to the names of the authors: Peter Spirtes and Clark Glymour) is the first and significant enhancement of the SGS algorithm, where the main motivation in the branch of constraint-based methods of reducing the amount of performed statistical tests for conditional independence, in particular these less reliable, has arose.

On the entry of both the algorithms SGS and PC we have a dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$. In the case when there exists the generative distribution $\mathcal{P}$ of the sample $\mathcal{D}$ both the algorithms, assuming that they receive the perfect knowledge about conditional independencies occurring withing the set of variables $\mathcal{V}$ (which is not true of course) return the Markov equivalence class of perfect maps of distribution $\mathcal{P}$. They determine two main characterizations of this class: the skeleton and set of v-structures.

The SGS and PC algorithm are in fact not only providing the characterization of the Markov equivalence class, but also returning an exemplary DAG belonging to this class. We will not go into details of this procedure, as it is in fact a standard solution used also in many other constraint-based algorithms.

The most interesting part of the algorithms in the branch of constraint-based learning is related to determining the skeleton and v-structures—simply because here lie the main and most significant differences of these algorithms. Thus we will present these two, as well as any other constraint-based Bayesian network learning method only till the step of obtaining the characterization of a Markov equivalence class. A reader interested in further steps related to converting this characterization into some concrete DAG should for example study the sections related to the SGS and PC algorithm included in the Spirtes et al. (2000) work.

4.2.2. Dealing with Uncertainty of Tests Results

In the realistic case, where the statistical tests are not giving us always a correct answer, both the SGS an PC methods still return some skeleton and set of v-structures, but the problem is that they will probably not represent the desired Markov equivalence class, and moreover they will possibly not even represent any Markov equivalence class—in the sense for example that every DAG consistent with the obtained skeleton and set of v-structures has some other v-structures too. There can appear even a bigger problem: the learned set of v-structures might determine a cycle in the graph. In such a case there is no even any DAG consistent with the returned by the algorithm skeleton and v-structures characterization.

This problem has also nowadays some standard solutions. Briefly telling, besides aforementioned algorithm extending some obtained Markov equivalence class characterization into a DAG, there are also performed some additional repairing procedures, related in particular to removing possibly appearing cycles. Again, we are not going into any details here—a reader interested in the topic might for example look at the Margaritis and Thrun (2000) work presenting some more advanced constraint-based learning solution, which will be the last topic of this chapter. Here an explicitly described exemplary repairing procedure can be found.

4.2.3. SGS—Description and Correctness Analysis

Algorithm 4.2.3.1 is the SGS algorithm. In practical implementation each conditional independence is examined with some statistical test for conditional independence, applied on the available for us sample $\mathcal{D}$—for example with the χ^2 or G^2 test.

Algorithm 4.2.3.1 The *SGS* Bayesian network structure learning algorithm.

1. Let us initialize the skeleton $\mathcal{G}$ as a complete undirected graph on the vertex set $\mathcal{V}$.
2. For each set of two different vertices $\{X, Y\} \subseteq \mathcal{V}$, if there exists a subset $\mathbb{Z} \subseteq \mathcal{V} \setminus \{X, Y\}$ such that the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ holds, then remove the edge between X and Y from $\mathcal{G}$.
3. Let us initialize the set of v-structures $\mathcal{S}$ as an empty set.
4. For each $Z \in \mathcal{V}$ and for each set of two different vertices $\{X, Y\} \subseteq \mathcal{V} \setminus \{Z\}$ such that the pairs (X, Z) and (Y, Z) are connected by an edge in $\mathcal{G}$ while the pair (X, Y) is not connected and there is no subset $\mathbb{Z} \subseteq \mathcal{V} \setminus \{X, Y\}$ such that $Z \in \mathbb{Z}$ and the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ holds add to the set $\mathcal{S}$ the triple (X, Z, Y).
5. Return the characterization of the Markov equivalence class—the skeleton $\mathcal{G}$ and set of v-structures $\mathcal{S}$.

The theoretical correctness, that is the ability to return the desired Markov equivalence class of perfect maps given that such class exists and we obtain the perfect knowledge about the considered conditional independencies is very easy to prove in the case of this algorithm. It is in fact an immediate consequence of the previously mentioned remarks.

Namely, because in the desired Markov equivalence class all d-separations correspond exactly to the conditional independencies existing in the generative distribution, the correctness of determining the skeleton in Step 2 follows directly from Remark 2.4.4.1, while the correctness of determining the set of v-structures in Step 4 follows from Remark 2.4.4.2.

4.2.4. PC—Description and Correctness Analysis

Algorithm 4.2.4.1 is the enhancement of the SGS algorithm—the PC algorithm. The main difference lies in the significant reduction of the number of performed statistical tests, as well as the size of the conditional subsets of variables appearing in these tests. We will precisely expose this for a while.

The tests for conditional independence in Step 2, which aim is to find the independence evidence in order to remove any considered edge (X, Y), are here performed only for the conditional subsets included in $N_X^{\mathcal{G}} \setminus \{Y\}$ and $N_Y^{\mathcal{G}} \setminus \{X\}$, where $N_Z^{\mathcal{G}}$ means the set of neighbors

of the vertex Z in the graph $\mathcal{G}$ (more precisely: in the current version of this graph—as it is constantly updated and reduced during this step). The algorithm investigates the conditional subsets for all together remaining connected pairs of vertices in the order of their size: 0 , 1, 2 and so on. When the size of the conditional subsets reaches the maximal size of all current sets of neighbors the skeleton learning part is finished. As we will further see such strategy is both theoretically sound and results in the interesting performance properties.

The $Sepset(X, Y)$ is a constantly created during Step 2 assignment to pairs of different variables X and Y the subset of remaining variables conditioning on which X and Y are independent. For each pair of the vertices where edge between them has been removed in Step 2 such assignment is created, and then it is used in Step 4, which results in avoiding further searching there for conditional independencies with the assist of statistical tests and speeding up the process of creating the v-structures set.

Algorithm 4.2.4.1 The *PC* Bayesian network structure learning algorithm.

1. Let us initialize the skeleton $\mathcal{G}$ as a complete undirected graph on the vertex set $\mathcal{V}$.

2. Set $l := 0$. Repeat:

   - For each set of two different vertices $\{X, Y\} \subseteq \mathcal{V}$ connected by an edge in $\mathcal{G}$ if there exists a subset $\mathbb{Z} \subseteq N_X^{\mathcal{G}} \setminus \{Y\}$ or a subset $\mathbb{Z} \subseteq N_Y^{\mathcal{G}} \setminus \{X\}$ such that $|\mathbb{Z}| = l$ and the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ holds, then remove the edge between X and Y from $\mathcal{G}$ and record $Sepset(X, Y) := Sepset(Y, X) := \mathbb{Z}$
   - Set $l := l + 1$.

   until for each vertex $X \in \mathcal{V}$ it holds that $|N_X^{\mathcal{G}}| \leqslant l$.

3. Let us initialize the set of v-structures $\mathcal{S}$ as an empty set.

4. For each $Z \in \mathcal{V}$ and for each set of two different vertices $\{X, Y\} \subseteq \mathcal{V} \setminus \{Z\}$ such that the pairs (X, Z) and (Y, Z) are connected by an edge in $\mathcal{G}$ while the pair (X, Y) is not connected and $Z \notin Sepset(X, Y)$ add to the set $\mathcal{S}$ the triple (X, Z, Y).

5. Return the characterization of the Markov equivalence class—the skeleton $\mathcal{G}$ and set of v-structures $\mathcal{S}$.

The theoretical correctness of the PC algorithm is again easy to see. Namely let us notice that:

- Every edge removed from $\mathcal{G}$ during Step 2 does not occur in the characterization of the class of perfect maps of the generative distribution—as each such removal is based on the appropriate conditional independence of two connected so far variables.

- Every pair of variables not connected in the skeleton characterizing the aforementioned class will be removed during Step 2. Namely, according to Remark 2.4.4.3 each two different and not connected in this skeleton vertices X and Y are d-separated by the set of parents of X or the set of parents of Y in any perfect map belonging to the considered class. The corresponding conditional independences hold in the generative distribution. So in particular X and Y are conditionally independent in this distribution given some subset $\mathbb{Z}$ of variables of either the neighbors of X or the neighbors of Y in the skeleton characterizing the class of perfect maps. From the first point we know

that during Step 2 of the PC algorithm the skeleton characterizing the class of perfect maps is always a subgraph of the constantly reduced skeleton $\mathcal{G}$. So in particular the aforementioned subset $\mathbb{Z}$ is during all the execution of Step 2 included in one of the sets $N_X^{\mathcal{G}} \setminus \{Y\}$ or $N_Y^{\mathcal{G}} \setminus \{X\}$. This subset will be proceeded in the round of conditional subsets of size $l = |\mathbb{Z}|$, leading to the conditional independence detection and removing the edge—provided that the whole process of searching for the conditional independence was not successfully finished already in the previous rounds.

- The correctness of the v-structures detection process in Step 4 follows immediately from Remark 2.4.4.2.

4.2.5. Dealing with Neglected Tests

In the practical application of both algorithms SGS and PC, in the case when the Spirtes et al. (2000) convention of aborting unreliable tests is applied (see Subsection 2.2.8), the conditional dependence is automatically assumed whenever the test is aborted. The reason lies in the nature of these methods—they start from a complete graph and try to find for each edge a conditional independence which will be the evidence for removing this edge. In the case of lack of such evidence the edge remains. So the assumption of the conditional independence in the case of the insufficiently reliable tests would obviously lead us to removing all edges from the skeleton—whenever on the input there is a sufficiently large dataset in the sense of the number of attributes. Namely, assuming that we operate on the dataset where sufficiently large conditional subsets of variables are neglected according to the Spirtes et al. (2000) rule every edge for which the evidence of removing would not be found with the reliable tests would be automatically removed in the moment of reaching the first insufficiently reliable one.

4.2.6. SGS and PC Performance—Scenario ST_1 and SP_1

We will finish this section with a detailed analysis of the SGS and PC algorithms from the point of view of their efficiency and reliability. Namely we investigate further all the aspects and scenarios listed respectively in Subsections 4.1.2 and 4.1.3.

We start with the SGS analysis in the first theoretical scenario ST_1. In Step 2 we search for the conditional independence evidence for every pair of different vertices $\{X, Y\} \subseteq \mathcal{V}$. Thus we pessimistically proceed through all the subsets of remaining variables. As the result, the SGS procedure performs pessimistically the exponential number of conditional independence tests with respect to the number of attributes n. The time complexity of the approach is exponential with regard to n as well, and linear with regard to the number of rows m—as each of the performed tests is linear with regard to m (the complexity with regard to n does not depend on m, and vice versa). The pessimistic size of the conditional set is equal to $n-2$.

In the scenario ST_1 the PC algorithm has exactly the same properties as SGS. Although there is much effort taken in this algorithm into the reduction of the number of performed tests, we cannot in the general theoretical scenario without any further assumptions exclude the case that we search through exponentially many with regard to n conditional subsets. Note that the skeleton representing the class of faithful representations of the generative distribution can be even a clique—in such a case the searching through all subsets would definitely happen in Step 2.

In the case of the first practical scenario SP_1 the performance of both the SGS and PC methods is the same as their performance in the scenario ST_1. We do not assume in this scenario anything, and as the result there is always an uncontrolled risk that the search for

conditional independence evidences in Step 2 of the SGS and PC algorithms will proceed through exponential with regard to n number of conditional subsets.

4.2.7. SGS and PC Performance—Scenario ST_2

Unfortunately, in the second theoretical scenario ST_2 the SGS performance does not change. Namely, the main complexity in particular still remains in Step 2 in the case of searching for an evidence for removing the edge from the skeleton—that is searching for some conditional independence of the two connected nodes, given that this edge exists in the desired faithful representation. Obviously the evidence cannot be found in such case— so the search will proceed through all the possible subsets—exactly as it was in the case of the scenario ST_1. As the result, both the pessimistic number of performed tests for conditional independence, as well as the time complexity of the whole approach remain exponential with regard to the number of attributes n of the dataset. With regard to the number of rows m we have linear time complexity of the whole approach (the complexities with regard to m and n are again independent). Also, the pessimistic size of the conditional set remains equal to $n-2$.

The PC algorithm in contrary achieves much better performance in the ST_2 scenario. The crucial observation here is that Step 2 of the algorithm will be ended not further than at the round of subsets of size $l = k+1$, where k is the maximal degree of vertices in the skeleton characterizing the class of perfect maps. We have previously pointed out that if two vertices are not connected in this skeleton, then the conditional independence of them given some subset of variables contained in the set of neighbors of one of them in this skeleton occurs. This implies that such conditional subset has size at most k—so it will be found during Step 2 in one of the rounds before $l = k+1$—assuming that the evidence for removing the edge was not found earlier. At the round corresponding to $l = k+1$—assuming the procedure reaches it—the graph $\mathcal{G}$ is exactly a desired skeleton characterizing the class of perfect maps, so each vertex has in particular a degree at most k in $\mathcal{G}$, and in particular the termination condition for the repeat loop is satisfied. As the result the pessimistic number of performed statistical tests for conditional independence in Step 2, as well as in the whole PC procedure (in Step 4 we do not perform any tests like it was in the case of SGS), is:

$$2\binom{n}{2}\left[\binom{n-2}{0}+\binom{n-2}{1}+\binom{n-2}{k}\right],$$

that is $\mathcal{O}(n^{k+2})$. Each conditional independence test is based on the conditional subset of size bounded by k, so the time complexity of calculating it is $\mathcal{O}(m)$. Checking the repeat loop termination condition in Step 2 has time complexity $\mathcal{O}(n)$ and such checking is performed at most $k+1$ times. The time complexity of the execution of Step 4 is also $\mathcal{O}(n)$, as we need to search for each $Z \in \mathcal{V}$ for the pair of unconnected vertices included in $N_Z^{\mathcal{G}}$ which has size bounded by k, and for each such pair $\{X, Y\}$ we check whether $Z \notin Sepset(X, Y)$ in a constant time, as the set $Sepset(X, Y)$ has size bounded by k. So the time complexity of the whole PC procedure is $\mathcal{O}(mn^{k+2})$. The pessimistic size of the conditional set in performed statistical tests is, as we have already mentioned it, k.

4.2.8. SGS Performance—Scenario SP_2

In the case of the second practical scenario SP_2, thanks to the applied Spirtes et al. (2000) solution, there is performed the test for conditional independence investigating the relation

$Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ only provided that:

$$m \geqslant c|Dom_X||Dom_Y| \prod_{Z \in \mathbb{Z}} |Dom_Z|,$$

where c is some constant. The above condition can be true only provided that:

$$m \geqslant c2^{|\mathbb{Z}|+2},$$

as we consider only a nontrivial setting, where each of the variables has at least two values. The above inequality we can transform into:

$$|\mathbb{Z}| \leqslant \log_2 m - (\log_2 c + 2).$$

So what we for sure know is that any test of conditional independence corresponding to the conditional set of size greater than $\log_2 m - (\log_2 c + 2)$ will be not performed (equivalently greater than $\lfloor \log_2 m - (\log_2 c + 2) \rfloor$, as we operate on integer sizes)—instead the conditional dependence will be automatically assumed. For the sake of convenience let us denote $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$.

Let us first analyze the pessimistic number of statistical tests performed by SGS in this scenario. In Step 2 we need to search for the conditional independence evidence for every pair of different vertices $X, Y \subseteq \mathcal{V}$. But assuming that we are performing these statistical tests in the order from the least to the largest conditional sets, the whole procedure of searching can be instantly aborted whenever it riches to subsets of size greater than b_m, as in the case of all the remaining tests the automatic verdict will be conditional dependence, and as a consequence the edge between X and Y will not be removed from the skeleton. So we will in this step pessimistically perform

$$\binom{n}{2}\left[\binom{n-2}{0} + \binom{n-2}{1} + ... + \binom{n-2}{b_m}\right]$$

tests for conditional independence, that is $\mathcal{O}(n^{b_m+2})$. In Step 4 we need to search for each $Z \in V$ and each pair of different vertices $\{X, Y\} \subseteq \mathcal{V} \setminus \{Z\}$ for the evidence of conditional independence of X and Y given any subset of remaining vertices containing variable Z. Similarly as previously, the search procedure can be instantly stopped whenever we reach to conditional subsets of size greater than b_m—as in such case it is known from advance that we will add to the v-structures set the triple (X, Z, Y). So in this step we will pessimistically perform

$$n\binom{n-1}{2}\left[\binom{n-3}{0} + \binom{n-3}{1} + ... + \binom{n-3}{b_m-1}\right]$$

tests for conditional independence, so again $\mathcal{O}(n^{b_m+2})$ tests. Summarizing, the whole SGS method requires pessimistically performing $\mathcal{O}(n^{b_m+2})$ tests, that is $\mathcal{O}(n^{\lfloor \log_2 m - \log_2 c \rfloor})$ tests. The time complexity of performing each test is bounded by $\mathcal{O}(mb_m)$, so the time complexity of the SGS approach is here $\mathcal{O}(mb_m n^{b_m+2})$, that is $\mathcal{O}((m \log_2 m)n^{\lfloor log_2 m - \log_2 c \rfloor})$. It holds that $n^{log_2 m} = m^{log_2 n}$, so the time complexity is polynomial both with regard to m and n—although the exponent with regard to each of these two input dataset dimensions depends directly on the second dimension. The pessimistic size of the conditional set appearing in statistical tests is clearly equal to $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$.

4.2.9. PC Performance—Scenario SP_2

The PC performance in the case of the scenario SP_2 is in fact identical. Whenever the repeat loop in Step 2 reaches $l = b_m + 1$ it can be instantly stopped, as any new conditional independence evidence will not be found and any further considered edge will remain in the skeleton. So the pessimistic number of tests performed in the PC procedure is:

$$2\binom{n}{2}\left[\binom{n-2}{0} + \binom{n-2}{1} + ... + \binom{n-2}{b_m}\right],$$

that is $\mathcal{O}(n^{b_m+2})$, exactly as it was in the case of SGS. The time complexity properties and the pessimistic size of the conditional set in performed statistical tests remain for the scenario SP_2 the same as in the case of SGS. In the case of the time complexity, in Step 2 we have in PC an additional cost of checking the loop termination condition—we check it $\mathcal{O}(b_m)$ times, each of them in $\mathcal{O}(n)$ time—so totally it costs us $\mathcal{O}(b_m n)$. Another additional cost in this step is related to storing the conditional subsets in the *Sepset* table. We create this table in $\mathcal{O}(n^2)$ time and update it in $\mathcal{O}(b_m)$ time whenever each next test for conditional independence in Step 2 is performed—as each element of the *Sepset* array has size bounded by b_m. However the cost of performing each such test is pessimistically $\mathcal{O}(mb_m)$, so the additional *Sepset* updating cost is entailed in it. The conclusion is that the overall time complexity of Step 2 remains the same as the complexity of SGS: $\mathcal{O}(mb_m n^{b_m+2})$. The cost of Step 4 is pessimistically $\mathcal{O}(b_m n^3)$ (for each $Z \in \mathcal{V}$ we search for no more than $\mathcal{O}(n^2)$ pairs of remaining vertices $\{X, Y\}$ included in $N_Z^{\mathcal{G}}$, and for each such pair we check whether $Z \notin Sepeset(X, Y)$ in $\mathcal{O}(b_m)$ time, as the size of $Sepset(X, Y)$ is bounded by b_m). It is in practical applications impossible that $b_m = 0$ (it would require unrealistically small sample size m), so except such degenerated scenario this cost is also entailed in $\mathcal{O}(mb_m n^{b_m+2})$, which is an overall time complexity.

4.2.10. SGS versus PC Performance—Conclusions

Summarizing, the effort of the PC algorithm in the aspect of reduction comparing to SGS the amount of performed statistical tests of conditional independence has its benefits. The PC performance is the same as the SGS performance in the scenarios ST_1, SP_1 and SP_2, and it is significantly better comparing to SGS in the case of the scenario ST_2. Namely, the execution time is reduced from the exponential with regard to n in the case of SGS to the polynomial in the case of PC. Moreover, the pessimistic size of the conditional set in performed statistical tests for conditional independence is also reduced drastically—from the linear with regard to n in the case of SGS to the constant one in the case of PC. Thus, the great improvement is both with regard to the efficiency and reliability of the method. Although this difference is visible in the only one, theoretical scenario, its significance cannot be ignored.

4.3. Grow-Shrink

We present in this section the algorithm which has established a notable progress in the domain of constraint-based Bayesian network structure learning algorithms. Its main aim is to decompose the whole problem of learning the structure into the significantly smaller ones, basing on the mentioned in Subsection 2.2.9 Markov blanket concept. We first reveal the real nature of Markov blankets, placing them in the strong relation with the Bayesian networks field. Then we describe the algorithm and show its correctness. We discuss the way

of resolving neglected tests for conditional independence. We also analyze the method with regard to all aspects given in Subsection 4.1.2 and all scenarios given in Subsection 4.1.3. Finally we sketch some possible enhancements of the algorithm which in some sense have been realized in the future algorithms, in particular in one described in the next chapter.

4.3.1. Markov Blanket Concept Revisited

In Subsection 2.2.9 we have introduced the notion of Markov blanket. Let us recall that the Markov blanket for the given variable T belonging to the set of random variables $\mathcal{V}$ is a minimal subset of variables excluding T such that T is independent from all the remaining variables given this subset.

The Markov blanket notion can be considered in isolation from Bayesian networks. In particular we have already exposed the most straightforward application of the blankets—feature selection in the classification task of a given decision attribute. Even the Markov blankets learning from a given dataset can be, and in fact was for a long time, considered in isolation from Bayesian networks, with the purpose usually related to the mentioned feature selection. The most recognizable Markov blanket approximation algorithms from this era, like KS (Koller and Sahami, 1996) and K2MB (Cooper et al., 1997), are the heuristic approaches without the theoretical soundness guaranteed, at most partially exploiting the real nature of the blankets and concentrating only on the feature selection application.

Since the introduction in 1988 by Judea Pearl the Markov blanket notion its real potential has been practically untouched until the Margaritis and Thrun (2000) work. This short article is without any doubts a milestone in the research related to the constraint-based Bayesian network structure learning. A very good comment about the previous efforts in this field has been given by the authors in this article:

The concept of the Markov blanket (...) is not new. (...) It is surprising, however, how little attention it has attracted for all its being a fundamental property of Bayesian nets.

The article is very briefly describing and analyzing the proposed algorithms. A much deeper analysis, in particular containing the complete proof of the theoretical soundness of the proposed methodology and the complete explanation of all the implementation details can be found in the PhD thesis of one of the coauthors—Dimitris Margaritis (Margaritis, 2003).

The following in fact well known since the genesis of Markov blankets fact is the main tool employed in the Margaritis and Thrun (2000) work. This result provides a strong connection between the Markov blanket and Bayesian network notion. The proof is simple, it can be found for example in the Neapolitan (2003) book.

Theorem 4.3.1.1 *Assume that the joint probability distribution $\mathcal{P}$ of some set of random variables $\mathcal{V}$ admits a faithful representation. Let a DAG $\mathcal{G}$ operating on the set of vertices $\mathcal{V}$ be an arbitrary chosen perfect map of the distribution $\mathcal{P}$. Then for every variable $T \in \mathcal{V}$ there exists exactly one Markov blanket of T with regard to $\mathcal{P}$. It consists of the variables corresponding to the set of parents, children, and parents of children of T in $\mathcal{G}$.*

It is important to point out that in the above statement it does not matter which perfect map we choose. Although separately considered the set of parents or the set of children of the vertex T can vary for different perfect map instances—the point of interest, that is the together taken set of parents, children and parents of children remains the same for any

perfect map. The reason is that all perfect maps are in one Markov equivalence class, so they all have the same skeleton and the same set of v-structures. As a consequence, Theorem 4.3.1.1 can be reformulated as follows:

Theorem 4.3.1.2 *Assume that the joint probability distribution $\mathcal{P}$ of some set of random variables $\mathcal{V}$ admits a faithful representation. Let $\mathcal{C}$ be the Markov equivalence class of all perfect maps of $\mathcal{P}$. Then for every variable $T \in \mathcal{V}$ there exists exactly one Markov blanket of T with regard to the distribution $\mathcal{P}$. It consists of all such variables $X \in \mathcal{V}$ that either T and X are connected by an edge in the skeleton characterizing $\mathcal{C}$, or there exists such $Z \in \mathcal{V}$ that the triple (X, Z, T) is one of the v-structures characterizing $\mathcal{C}$.*

4.3.2. Two Algorithms in One

There are to main results of the Margaritis and Thrun (2000)—two algorithms, both having common name: Grow-Shrink (GS):

- The first algorithm is a Markov blanket learning method. On the entry of the algorithm is some dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ and one chosen attribute $T \in \mathcal{V}$. As usually, we treat the given dataset as a sample where the attributes in $\mathcal{V}$ correspond to some random variables. The method returns a subset of the set of variables $\mathcal{V} \backslash \{T\}$ which is the learned Markov blanket of T. In general there might be more than one such Markov blanket, while the GS method returns always one subset. Its purpose corresponds to the scenario of the generative distribution admitting a faithful representation, where according to Theorems 4.3.1.1 and 4.3.1.2 the Markov blanket is guaranteed to be unique.

  GS was the first Markov blanket learning algorithm in the history which was theoretically sound. This means that it is the first algorithm which returns the correct, unique solution $MB_T^{\mathcal{P}}$ (the notation introduced in Definition 2.2.9.1) in the case when there exists the generative distribution $\mathcal{P}$ of the sample which admits a faithful representation, and there can be retrieved a 100 percent correct information about the conditional independencies occurring in $\mathcal{P}$.

- The second algorithm is a constraint-based Bayesian network structure learning algorithm, which uses as a subroutine the first Markov blanket learning algorithm. We will describe further in details both these approaches. Let us now only mention that the introduced here structure learning algorithm has opened a completely new application of Markov blankets. So far their applications were in fact limited to the feature selection task. Here Markov blankets learning of each variable in $\mathcal{V}$ has become an initial preprocessing step, on the basis of which the network structure learning is then performed. This new application of blankets is very successful, as we will see in further analysis of the GS algorithm, especially comparing it to the previously discussed SGS and PC algorithms.

  But the implications of this new Markov blankets application have gone even further. In fact the authors could not even expect that their results will become an inspiration in the future for even more successful solutions—which are nowadays recognized as a branch of local learning algorithms. In further sections we will see the few most notable solutions in this branch. The root of all this great progress in the constraint-based Bayesian network learning algorithms is placed exactly here, in the GS mechanism.

4.3.3. Markov Blanket Learning—Description and Correctness Analysis

Algorithm 4.3.3.1 is the Grow-Shrink algorithm understood as the Markov blanket learning subroutine. The name of the algorithm is easily interpretable here, as the method consists of two main phases, where in the first phase (Grow) we potentially add variables to the initially empty set $\mathbb{MB}$ which at the end of the whole method is returned as a Markov blanket of T, while in the second phase (Shrink) there are potentially removed some of the elements from $\mathbb{MB}$.

Algorithm 4.3.3.1 The GS Markov blanket learning algorithm.

1. Let us initialize the set $\mathbb{MB}$ representing the Markov blanket $MB_T^{\mathcal{P}}$ as an empty set.
2. (Grow phase) Repeat two times:
   - One time proceed each variable in $\mathcal{V} \setminus \{T\}$, in any order. If for the currently considered variable X the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ does not hold add X into the set $\mathbb{MB}$.
3. (Shrink phase) One time proceed each variable belonging after the Grow phase to the set $\mathbb{MB}$, in any order. If for the currently considered variable X the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB} \setminus \{X\})$ holds remove X from the set $\mathbb{MB}$.
4. Return the set $\mathbb{MB}$.

It is easy to see that this method is theoretically sound. Namely, when there exists a generative distribution $\mathcal{P}$ of an input sample admitting a faithful representation and $Ind_{\mathcal{P}}$ relations are resolved always correctly, then:

- After the Grow phase there holds $MB_T^{\mathcal{P}} \subseteq \mathbb{MB}$. In order to see this let us consider any perfect map $\mathcal{G}$ for the distribution $\mathcal{P}$. According to Theorem 4.3.1.1 it is sufficient to show that each neighbor of T and each parent of any children of T in $\mathcal{G}$ is included in $\mathbb{MB}$.

  After the first processing of each variable in the Grow phase each neighbor X of T in the graph $\mathcal{G}$ is included in $\mathbb{MB}$—because according to Remark 2.4.4.1 the vertices T and X are not d-separated by any subset of remaining nodes in $\mathcal{G}$—so the corresponding conditional independence of them in $\mathcal{P}$ cannot also occur, as $\mathcal{G}$ is a perfect map of $\mathcal{P}$.

  After the second processing of each variable in the Grow phase each parent X of any children Z of the node T in the graph $\mathcal{G}$ is included in $\mathbb{MB}$—because the vertex Z already belongs to $\mathbb{MB}$, and according to Remark 2.4.4.2 the vertices T and X are not d-separated by any subset of remaining nodes containing Z in $\mathcal{G}$—so the corresponding conditional independence also cannot occur.

- After the Shrink phase there holds $MB_T^{\mathcal{P}} = \mathbb{MB}$. The explanation can be following. First of all, any currently processed variable X belonging to $MB_T^{\mathcal{P}}$ will be not removed in the Shrink phase simply because the analogical considerations as in the previous point shows that in any perfect map for $\mathcal{P}$ the d-separation of X and T given all remaining variables in current M—thus in particular all remaining variables in $MB_T^{\mathcal{P}}$, cannot hold, so the corresponding conditional independence in $\mathcal{P}$ also cannot appear. Secondly, directly from the definition of the Markov blanket for any subset $\mathbb{S} \supseteq MB_T^{\mathcal{P}}$ we have that $Ind_{\mathcal{P}}(T, \mathbb{S} \setminus MB_T^{\mathcal{P}} \mid MB_T^{\mathcal{P}})$ holds—so in particular for any $X \in \mathbb{S} \setminus MB_T^{\mathcal{P}}$

there holds $Ind_{\mathcal{P}}(T, X \mid \mathbb{S} \setminus \{X\})$—which means that any processed variable X outside of the $MB_T^{\mathcal{P}}$ in the Shrink phase will be removed from $\mathbb{MB}$.

4.3.4. Network Structure Learning—Description and Correctness Analysis

Algorithm 4.3.4.1 is the GS Bayesian network structure learning algorithm. As it can be noticed it is a bit similar approach to SGS, although some interesting differences can be noticed. First of all, we start from an empty skeleton $\mathcal{G}$, not the complete one as in SGS, and instead of trying to find the evidence for removing each edge we try to find here the evidence for inserting each edge. There is in particular suitable here an opposite strategy of neglecting the statistical tests according to the Spirtes et al. (2000) convention—we will discus it later. Moreover, there is one crucial preprocessing step here comparing to SGS—we search for the Markov blanket of each variable first. In order to determine them we can use for example the introduced in previous subsection Algorithm 4.3.3.1. Then the search for conditional independencies in two steps related to respectively constructing the skeleton and set of v-structures is here not performed among all possible subsets of remaining variables, but only among the subsets contained in the appropriate one Markov blanket. It is easy to imagine what such modification can give. In the case when the blankets have limited size the complexity of such searching for an evidence is radically reduced comparing to the mentioned SGS. The reliability of obtained results can be in this scenario also much better in the case of GS, as the conditional subsets have limited size. We will return to this aspect later, when we analyze formally the performance of the GS algorithm according to all aspects and scenarios defined in Section 4.1.

Algorithm 4.3.4.1 The GS Bayesian network structure learning algorithm.

1. For each $X \in \mathcal{V}$ determine the set $M(X)$ representing its learned Markov blanket $MB_X^{\mathcal{P}}$ using any Markov blanket learning algorithm, for example Algorithm 4.3.3.1.
2. Let us initialize the skeleton $\mathcal{G}$ as an empty undirected graph on the vertex set $\mathcal{V}$.
3. For each $X \in \mathcal{V}$ and $Y \in M(X)$ connect with an edge nodes X and Y in $\mathcal{G}$ if $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S})$ does not hold for any $\mathbb{S} \subseteq \mathbb{Z}$, where $\mathbb{Z}$ is the smaller set (or any of the two if their size is the same) among the pair of sets: $M(X) \setminus \{Y\}$ and $M(Y) \setminus \{X\}$.
4. Let us initialize the set of v-structures $\mathcal{S}$ as an empty set.
5. For each $Z \in \mathcal{V}$ and for each set of two different vertices $\{X, Y\} \subseteq \mathcal{V} \setminus \{Z\}$ such that the pairs (X, Z) and (Y, Z) are connected by an edge in $\mathcal{G}$ while the pair (X, Y) is not connected add the triple (X, Z, Y) to the set $\mathcal{S}$ if $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S} \cup \{Z\})$ does not hold for any $\mathbb{S} \subseteq \mathbb{Z}$, where $\mathbb{Z}$ is the smaller set (or any of the two if their size is the same) among the pair of sets: $M(X) \setminus \{Y\}$ and $M(Y) \setminus \{X\}$.
6. Return the characterization of the Markov equivalence class—the skeleton $\mathcal{G}$ and set of v-structures $\mathcal{S}$.

Algorithm 4.3.4.1 is theoretically sound as well—assuming that we use in Step 1 Algorithm 4.3.3.1 or any other sound Markov blanket learning algorithm. It returns the characterization of the Markov equivalence class of perfect maps of the generative distribution $\mathcal{P}$—assuming it exists and $Ind_{\mathcal{P}}$ relations are always resolved correctly:

- Thanks to the soundness of the applied Markov blanket learning subroutine subject to the above assumptions in Step 1 we have that $M(X) = MB_X^{\mathcal{P}}$ for all $X \in \mathcal{V}$.

- In Step 3 we should notice first of all that for any variable X we consider only these variables Y as a potential neighbors of X in $\mathcal{G}$ which satisfy $Y \in M(X)$, that is $Y \in MB_X^{\mathcal{P}}$ subject to our theoretical scenario. Such limitation of considered edges is correct—as in the case when $Y \notin MB_X^{\mathcal{P}}$ according to Theorem 4.3.1.2 we have that X and Y are not neighbors in the skeleton characterizing the class of perfect maps.

  According to Remark 2.4.4.3 in any perfect map $\mathcal{H}$ for $\mathcal{P}$ each two nodes X and Y are not connected by an edge if and only if they are d-separated by one of the two sets: $\pi_X^{\mathcal{H}}$ or $\pi_Y^{\mathcal{H}}$. In this statement we can replace the d-separation relation with the conditional independence relation $Ind_{\mathcal{P}}$, as we talk about the faithful representation. Moreover, according to the characterization of the Markov blanket given in Theorem 4.3.1.1 each Markov blanket contains in particular all parents of the target node in any perfect map, so we can express the above fact on the higher level of abstraction as follows: X and Y are not connected in the skeleton characterizing the class of perfect maps if and only if the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S})$ for some $\mathbb{S} \subseteq MB_X^{\mathcal{P}} \setminus \{Y\}$ or $\mathbb{S} \subseteq MB_Y^{\mathcal{P}} \setminus \{X\}$. But something stronger can be proved, namely it is sufficient to search for the conditional independence evidence for the case of conditional subsets $\mathbb{S}$ included in one, arbitrary chosen set: either $MB_X^{\mathcal{P}} \setminus \{Y\}$ or $MB_Y^{\mathcal{P}} \setminus \{X\}$. We formally state this fact below, in the language of d-separations, in Theorem 4.3.4.1. The proof can be found in the Margaritis (2003) PhD dissertation. That is why in Step 3 the subsets of one of these two sets are examined—and of course the smaller one is chosen in order to improve the efficiency of the approach.

  There might arise one question regarding the considered here Step 3. Why do we consider each pair of different vertices (X, Y) pessimistically two times (one time as a pair: X and $Y \in M(X)$ and second time as a pair Y and $X \in M(Y)$), instead of proceeding each such pair one time, as it was in the case of SGS and PC algorithms ? The reason is that although in the assumed here theoretical conditions there is a perfect symmetry: $X \in MB_Y^{\mathcal{P}} \iff Y \in MB_X^{\mathcal{P}}$ (which is an instant consequence of Theorem 4.3.1.1 or 4.3.1.2), in practice this property does not necessarily hold for the learned blankets $M(X)$ and $M(Y)$. As the result only one of the two inclusions can potentially hold—so in order to manage such cases all suitable ordered pairs of vertices are processed.

- In Step 5 it would be sufficient to check for each candidate to be a v-structure triple (X, Z, Y) whether Z is included in the subset $\mathbb{S}$ such that $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S})$ holds. According to Remark 2.4.4.2, if Z does not belong to such subset $\mathbb{S}$ then (X, Z, Y) is a v-structure, otherwise no. Moreover, there would be also possibility to neglect all such triples (X, Z, Y) for which there holds $X \notin M(Y)$ and $Y \notin M(X)$—as according to Theorem 4.3.1.2 there is no possibility in such case that the triple (X, Z, Y) is a v-structure. For all the remaining considered here pairs of X and Y we have already found in Step 3 the subset $\mathbb{S}$ such that $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S})$ holds. Thus we could apply here the *Sepset* array technique employed in the PC algorithm.

  In the original GS there is applied however another procedure—more time expensive, but also more reliable. It can be seen exactly in Step 5 of Algorithm 4.3.4.1. Namely, thanks to this that we perform here searching for the conditional independence evidence in somehow limited space of subsets contained in Markov blankets, the GS algorithm checks more exhaustively each candidate v-structure. It determines whether there is no any conditional independence of X and Y given any subset $\mathbb{S}$ included in the considered

blanket with an additional element Z. Only in the case when such independence is not found the processed triple is added to the v-structures set $\mathcal{S}$. This of course does not affect the theoretical soundness of the approach, but in practice where in particular we have an imperfect knowledge about the $Ind_{\mathcal{P}}$ relation such procedure can lead to the situation where we are less often assigning any triple to the v-structures set $\mathcal{S}$, adding only these directions to the Markov blanket equivalence class characterization which are very reliable. This in particular might also mean less problems in the applied in practice and mentioned in the previous section additional repairing procedure assuring the consistency of the returned model.

Theorem 4.3.4.1 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary DAG, and let us consider two different nodes $X, Y \in \mathcal{V}$. Let $\mathbb{M}$ be the set of parents, children and parents of children of X in $\mathcal{G}$. Then it holds that X and Y are not connected by an edge in $\mathcal{G}$ if and only if the d-separation $Ind_{\mathcal{G}}(X, Y \mid \mathbb{M} \setminus \{Y\})$ occurs.*

4.3.5. Dealing with Neglected Tests

As we have already noticed it, the GS algorithm methodology is from one side similar to SGS, but in some aspects it stands on the complete opposite side. In particular, in contrary to SGS, in GS we build a skeleton from an empty graph. Moreover, in the Markov blanket learning subroutine we also start from an empty set representing a given blanket, and we try to find an evidence to include some variable there. This is especially important to note if we want to apply the Spirtes et al. (2000) convention of neglecting the least reliable statistical tests for conditional independence. Here an opposite to SGS neglecting strategy is suitable.

Imagine that we would apply the SGS strategy in the case of the Markov blanket learning subroutine—Algorithm 4.3.3.1. Let us recall that this would mean that whenever the size of the conditional set in the investigated conditional independence relation exceeds some value depending on the dimension m and the number of values of the dataset attributes (according to Subsection 4.2.8 this value is at most equal to $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$, where c is a specified by a user constant—in the case of binary attributes) the statistical test is aborted and the conditional dependence is automatically returned. In the case of Algorithm 4.3.3.1 such strategy would be very dangerous—as whenever the size of the constructed there set $\mathbb{MB}$ in the Grow phase exceeds b_m, all the remaining unprocessed variables will be automatically included to $\mathbb{MB}$. What is even worse such obtained potentially huge set will not be reduced even by one variable in the Shrink phase if its size is at least $b_m + 2$—as in such case any requested conditional independence evidence will be neglected and conditional dependence will be assumed, due to a larger than b_m conditional set.

The much more sensible and indeed applied in practice choice in the case of GS is to automatically assume a conditional independence whenever the test is neglected. In such case Markov blankets cannot grow to some totally degenerated form. Moreover, note that after applying this strategy in Step 1 of Algorithm 4.3.4.1 there is no longer in Step 3 any problem regarding the neglected tests for conditional independence—simply because no such unreliable test can occur there. Namely, notice that each of the learned in Step 1 Markov blanket will have size at most $b_m + 1$ (as the last insertion of the variable to the set $\mathbb{MB}$ in the Grow phase of Algorithm 4.3.3.1 can occur when $\mathbb{MB}$ contains b_m elements)—which means that clearly in Step 3 of Algorithm 4.3.4.1 every conditional set have size at most b_m, thus it is performed in the usual manner. This is an important observation, because as we remember in the case of the SGS or PC algorithm the strategy of the automatic assumption of an independence in the case of too large conditional sets could lead to the total degeneration

of the result in the form of an empty skeleton. Here there is no risk of this degeneration when applying this strategy, as in Step 3 no tests with conditional sets exceeding b_m appears.

4.3.6. Performance—Analysis Scheme

Let us analyze now the performance of GS understood as Algorithm 4.3.4.1 with respect to all scenarios defined in Subsection 4.1.3 and in all aspects introduced in Subsection 4.1.2. As usually, we do it with regard to the input dataset, having m objects and n attributes. We will assume here that in Step 1 of Algorithm 4.3.4.1 we apply the GS Markov blanket learning subroutine, that is Algorithm 4.3.3.1. The main points of interest in Algorithm 4.3.4.1 are Steps:

- 1—performing the GS Markov blanket learning procedure for each variable,
- 3—determining the skeleton characterizing the class of perfect maps,
- 5—determining the set of v-structures characterizing this class.

We will perform the analysis separately for each of these three steps. It will be easy to derive from this detailed analysis the overall performance of the algorithm. As we will soon see the separated analysis of these three steps will have its specific aim in further sections.

4.3.7. Performance—Scenario ST_1 and SP_1

Let us start with the theoretical scenario ST_1. First of all, let us analyze the number of performed statistical tests for conditional independence:

- Step 1 in Algorithm 4.3.4.1 corresponds to n times performed Algorithm 4.3.3.1. In this algorithm we clearly perform $\mathcal{O}(n)$ tests, so overall in Step 1 we perform $\mathcal{O}(n^2)$ tests.
- Pessimistically the obtained in Step 1 Markov blankets can contain all remaining variables, so in Step 3 we pessimistically perform the exponential with regard to n number of tests.
- The same argumentation leads us to the conclusion that in Step 5 we pessimistically perform also the exponential with regard to n number of tests.

So overall the number of performed tests in Algorithm 4.3.4.1 is pessimistically exponential with regard to n. Now let us consider the time complexity aspect:

- The time complexity of Algorithm 4.3.3.1 is $\mathcal{O}(mn^2)$, as we perform there $\mathcal{O}(n)$ tests, where each of them costs $\mathcal{O}(mn)$ time. Actually, it is possible to implement the Grow phase in such way that its time complexity is $\mathcal{O}(mn)$—by appropriately storing the conditional frequencies corresponding to the currently calculated test for conditional independence and quickly updating this structure in the moment of performing the next test. Unfortunately, such improvement can not be achieved in the Shrink phase. So the time complexity of Step 1 of Algorithm 4.3.4.1 is $\mathcal{O}(mn^3)$.
- The time complexity of Step 3 and 5 is linear with regard to m and exponential with regard n (the complexity with regard to m does not depend on n and vice versa).

Overall we have that the time complexity of Algorithm 4.3.4.1 is (independently) linear with regard to m and exponential with regard to n. Finally let us consider the pessimistic size of the conditional set in performed tests:

- Pessimistically all processed variables can be included to the set $\mathbb{MB}$ in the Grow phase of Algorithm 4.3.3.1. So the pessimistic size of the conditional set in Step 1 of Algorithm 4.3.4.1 is equal to $n-2$.

- In Step 3 and 5 we pessimistically can deal with the Markov blankets containing all variables—so here also the pessimistic size of the conditional set is $n-2$.

So the pessimistic size of the conditional set in Algorithm 4.3.4.1 is $n-2$.

Summarizing, we have obtained that in the first theoretical scenario ST_1 all three algorithms: SGS, PC and GS have exactly the same performance, in all aspects.

In the case of the first practical scenario SP_1, where we do not assume anything, it is easy to check that the GS performance in all aspects remains the same as in the scenario ST_1. Moreover, the performance of each of Steps: 1, 3 and 5 of Algorithm 4.3.4.1 remains unchanged in all aspects. In the scenario SP_1, similarly as in ST_1, all three considered so far methods: SGS, PC and GS have the same performance in all aspects.

4.3.8. Performance—Scenario ST_2

Let us now consider the second theoretical scenario ST_2, where we assume that each node in the faithful representation of the generative distribution has degree at most k. Let us notice that in such case the maximal possible size of any Markov blanket is equal to k^2, as according to the characterization given in Theorem 4.3.1.1 the most pessimistic scenario is when the Markov blanket corresponds to k children of the considered variable, together with $k-1$ parents of each child. We start the analysis with the investigation of the number of performed statistical tests for conditional independence:

- Regardless of the limited size of Markov blankets in Algorithm 4.3.3.1 what we know for sure is that in the Grow phase we perform the test for conditional independence for each proceeded variable—thus we perform here $\mathcal{O}(n)$ tests. So in Step 1 of Algorithm 4.3.4.1 we perform $\mathcal{O}(n^2)$ tests.

- In Step 3 we have that for each $X \in \mathcal{V}$ $|M(X)| \leqslant k^2$, which means that in this step we perform only $\mathcal{O}(n)$ tests. Namely, for each $X \in \mathcal{V}$ we have to consider at most k^2 variables $Y \in M(X)$, and for each of them we perform the limited number of tests corresponding to all conditional subsets included in one of the Markov blankets, which has size bounded by k^2.

- Similarly, in Step 5 we perform only $\mathcal{O}(n)$ tests. For each variable $Z \in \mathcal{V}$ we search for the candidate to be a v-structure triple (X, Z, Y) only among X and Y belonging to the set of neighbors of Z in the already created in Step 3 skeleton $\mathcal{G}$. This set of neighbors has size bounded by k^2, so we proceed through the limited number of candidate triples. For each such triple we perform the limited number of tests for conditional sets corresponding to all subsets included in one limited by k^2 Markov blanket.

So totally in Algorithm 4.3.4.1 we perform $\mathcal{O}(n^2)$ tests. Now let us consider the time complexity aspect:

- Similarly as in the scenario ST_1 and SP_1 we obtain here that the time complexity of Step 1 of Algorithm 4.3.4.1 is $\mathcal{O}(mn^3)$. Although the Grow phase of Algorithm 4.3.3.1 can be performed in $\mathcal{O}(mn)$ time, there is no any guarantee regardless of the theoretical assumptions here that on the entry of the Shrink phase the set $\mathbb{MB}$ will have bounded anyhow size, so the Shrink phase costs still pessimistically $\mathcal{O}(mn^2)$ time.

- In Step 3 and 5 we perform $\mathcal{O}(n)$ tests, and each of these tests corresponds to the conditional set included in some Markov blanket—so of the limited size (this is not important here, however it is easy to note that also in the case of Step 5 not only set $\mathbb{S}$ belongs to one of the sets $M(X) \setminus \{Y\}$ or $M(Y) \setminus \{X\}$, but also the additional element Z—which actually belongs to both of these sets, as it is the neighbor of X and Y in the obtained in Step 3 skeleton $\mathcal{G}$). As a consequence each performed here test costs only $\mathcal{O}(m)$ time. As the result the time complexity of both Steps 3 and 5 is $\mathcal{O}(mn)$.

Summarizing the above points, the time complexity of Algorithm 4.3.4.1 is $\mathcal{O}(mn^3)$. What remains to analyze is the pessimistic size of the conditional set in performed statistical tests for conditional independence:

- In Step 1 of Algorithm 4.3.4.1 the pessimistic size is equal to $n-2$, as pessimistically we can add even all processed variables to the set $\mathbb{MB}$ in the Grow phase of an auxiliary procedure—Algorithm 4.3.3.1.

- In Step 3 any performed test investigating the existence of an edge between nodes X and Y corresponds to the conditional set included in the smaller of the sets $M(X) \setminus \{Y\}$ and $M(Y) \setminus \{X\}$. We have that $Y \in M(X)$ so the size of the smaller set for sure does not exceed $k^2 - 1$. As the result the pessimistic size of the conditional set in this step is $k^2 - 1$.

- In Step 5 for any candidate to be a v-structure triple of variables (X, Z, Y) the maximal possible considered conditional set corresponds to the set $\mathbb{Z} \cup \{Z\}$ where $\mathbb{Z}$ is the smaller of the sets $M(X) \setminus \{Y\}$ and $M(Y) \setminus \{X\}$. Here there is no any guarantee that $Y \in M(X)$ or $X \in M(Y)$, but for sure $Z \in \mathbb{Z}$, which we have already explained in the last point of the time complexity analysis. So the pessimistic size of the conditional set in Step 5 is k^2.

We have obtained that the pessimistic size of the conditional set in performed tests in Algorithm 4.3.4.1 is $n - 2$.

As we can see, in the scenario ST_2 there can be observed very significant differences between the three considered so far methods: SGS, PC and GS:

- With respect to the number of performed tests and time complexity the undisputed winner is GS - it requires performing only $\mathcal{O}(n^2)$ tests and costs only $\mathcal{O}(mn^3)$ time. For comparison, let us remind that the PC algorithm requires performing $\mathcal{O}(n^{k+2})$ tests and has the time complexity $\mathcal{O}(mn^{k+2})$, while in the case of the SGS method both the pessimistic number of performed tests and the time complexity are exponential with regard to n (the time complexity is linear with regard to m).

- The most reliable method is PC, as its pessimistic size of the conditional set in performed tests is only k, while in the case of both the SGS and GS methods this pessimistic size is equal to $n - 2$.

4.3.9. Performance—Scenario SP_2

The second practical scenario SP_2 is the most interesting one here. Let us analyze now the GS method with regard to it. In this scenario there is only one assumption: whenever the considered conditional independence relation corresponds to the conditional set of size larger than $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$, the statistical test for conditional independence is not performed and—according to what we have previously explained—the conditional

independence is automatically assumed. As usually we start with the investigation of the number of performed tests for conditional independence:

- In the auxiliary Algorithm 4.3.3.1 we pessimistically perform in the Grow phase the test for each processed variable. Such situation can occur when during this processing the size of the set $\mathbb{MB}$ does not exceed b_m. This set can in fact not grow—the only required condition is that for each processed for inclusion variable the corresponding conditional independence test returns an independence. Thus pessimistically we perform in this algorithm $\mathcal{O}(n)$ tests, and so we perform $\mathcal{O}(n^2)$ tests in Step 1 of Algorithm 4.3.4.1.

- Note that after performing Step 1 for each $X \in \mathcal{V}$ there holds $|M(X)| \leqslant b_m + 1$, because the set $\mathbb{MB}$ in Algorithm 4.3.3.1 can achieve maximally the size $b_m + 1$. Any more variable cannot be included to $\mathbb{MB}$ when it achieves this size, as according to the applied Spirtes et al. (2000) convention the conditional independence test performed during processing any further variable will be neglected and the conditional independence will be assumed.

  So for each $X \in \mathcal{V}$ we consider no more than $b_m + 1$ candidate neighbors $Y \in M(X)$, and for each such Y we perform at most one test for each subset of the smaller of the sets $M(X) \setminus \{Y\}$ and $M(Y) \setminus \{X\}$. We have that $Y \in M(X)$, so we perform for each Y at most 2^{b_m} tests. Summarizing, in Step 3 we perform $\mathcal{O}(nb_m 2^{b_m})$ tests, that is $\mathcal{O}((m \log_2 m)n)$ tests.

- In Step 5 for each $Z \in \mathcal{V}$ we consider at most $\binom{b_m+1}{2}$, that is $\mathcal{O}(b_m^2)$, pairs of different and not connected by an edge neighbors X and Y of Z, and for each corresponding to such pair triple (X, Z, Y) we are performing at most 2^{b_m} tests corresponding to all subsets containing Z of the smaller of the sets $M(X)\setminus\{Y\}$ and $M(Y)\setminus\{X\}$ (there is no guarantee that $Y \in M(X)$ or $X \in M(Y)$). So totally we perform in Step 5 $\mathcal{O}(nb_m^2 2^{b_m})$ tests, that is $\mathcal{O}(m(\log_2 m)^2 n)$ tests.

Summarizing, we have obtained that the pessimistic number of performed statistical tests for conditional independence in Algorithm 4.3.4.1 is $\mathcal{O}(m(\log_2 m)^2 n + n^2)$. Now let us consider the time complexity aspect:

- As we already know in Step 1 of Algorithm 4.3.4.1 we perform $\mathcal{O}(n^2)$ tests. Each of these tests costs only $\mathcal{O}(mb_m)$ time, that is $\mathcal{O}(m \log_2 m)$—as the maximal possible size of the conditional set appearing in these tests is b_m. So the time complexity of Step 1 is $\mathcal{O}((m \log_2 m)n^2)$.

- In Step 3 we perform $\mathcal{O}((m \log_2 m)n)$ tests, and thanks to the limited by $b_m + 1$ Markov blankets each such test costs $\mathcal{O}(mb_m)$ time, that is $\mathcal{O}(m \log_2 m)$. So the time complexity of Step 3 is $\mathcal{O}(m^2(\log_2 m)^2 n)$.

- In Step 5 we perform $\mathcal{O}(m(\log_2 m)^2 n)$ tests, so the time complexity of this step is $\mathcal{O}(m^2(\log_2 m)^3 n)$.

The time complexity of the whole Algorithm 4.3.4.1 has turned out to be $\mathcal{O}((m \log_2 m)n^2 + m^2(\log_2 m)^3 n)$. What remains to analyze is the pessimistic size of the conditional set in performed tests. What is for sure known is that it cannot exceed b_m. It turns out that it is exactly equal to b_m in all steps, as:

- In Step 1 of Algorithm 4.3.4.1 this pessimistic size is clearly equal to b_m.

- In Step 3 and 5 we have that the size of $M(X)$ for all $X \in \mathcal{V}$ is bounded by $b_m + 1$, so the pessimistic size of conditional sets is not decreased and remains b_m.

The conclusion is that the performance of the GS algorithm is in the case of the last practical scenario SP_2 significantly better than the performance of the SGS and PC algorithms:

- GS requires to perform only $\mathcal{O}(m(\log_2 m)^2 n + n^2)$ tests for conditional independence, while both SGS and PC pessimistically use $\mathcal{O}(n^{\lfloor \log_2 m - \log_2 c \rfloor})$ tests. As we have noticed it in the previous section, this complexity in the sense of the number of performed tests of SGS and PC is polynomial with regard to both m and n, however the polynomial with regard to one of this input dataset dimensions depends directly on the second dimension. A more favorable is the complexity of GS, where the method is polynomial with regard to both m and n independently.

- An analogical situation we have in the comparison with respect to the time complexity—as the time complexity of GS is $\mathcal{O}((m \log_2 m)n^2 + m^2(\log_2 m)^3 n)$, while the time complexity of both SGS and PC is $\mathcal{O}((m \log_2 m)n^{\lfloor \log_2 m - \log_2 c \rfloor})$.

- Only with respect to the pessimistic size of the conditional set in performed tests all three methods are equal—the common pessimistic size for them is b_m.

4.3.10. Summary and Possible Enhancements

The Markov blanket approximation has turned out to be a very useful step supporting the constraint-based Bayesian network learning methods. Margaritis and Thrun (2000) made an important contribution in this area. The GS algorithm is without doubts a more efficient solution than the previous basic constraint-based approaches, like SGS and in particular PC. This advantage we have clearly seen in two scenarios ST_2 and SP_2, where the key property of GS is that the initially learned Markov blankets, within which in fact all further search for the optimal network structure is conducted, have appropriately limited size.

The only one aspect in which GS is weaker than the preceding algorithms is the reliability in the case of the scenario ST_2, where the PC method is the undisputed winner. Although the time efficiency of PC ($\mathcal{O}(mn^{k+2})$) is here in general much worse than GS ($\mathcal{O}(mn^3)$)—the significantly smaller pessimistic size of the conditional set (reduction from $n-2$ to k) has its value, and so in the specific applications the PC algorithm might be more desirable.

Several enhancements of the GS algorithm are possible. Some of them are presented in the Margaritis (2003) PhD thesis. One of the possible modifications can be applied at the beginning of an auxiliary GS Markov blanket learning subroutine—that is in Algorithm 4.3.3.1. The idea is that we can somehow enhance both the reliability and efficiency of this algorithm. This enhancement does not cause any improvements which would be visible in the performed above complex analysis of GS in the four considered scenarios—but in practice it might cause indeed a difference in both the quality of the result and execution time.

The idea is to apply at the beginning of Algorithm 4.3.3.1 an additional heuristic ordering of all further processed variables. Namely, we can arrange the variables from the set $\mathcal{V} \setminus \{T\}$ in the descending order with regard to the function $Str_{\mathcal{P}}(X) = Assoc_{\mathcal{P}}(X, T \mid \emptyset)$—where the $Assoc_{\mathcal{P}}$ function reflects the conditional association of considered variables. Here the conditional set is empty, thus we are interested in the measurement of a direct dependence between each processed variable and T. The $Assoc_{\mathcal{P}}$ function itself can be defined as we proposed it in Subsection 2.2.6 in Equation 2.2.6.1.

The obtained order is used in the Grow phase, where we process the variables exactly according to it. The idea standing behind such additional heuristic ordering of processed

variables is very natural—we simply want to consider for the inclusion to the set $\mathbb{MB}$ these variables for which there is a biggest chance that they indeed should belong to the learned Markov blanket. If we place in the set $\mathbb{MB}$ very early all variables indeed belonging to the Markov blanket of the target variable T, then this superset of the blanket is small and will not grow any more, as all remaining unprocessed variables will be conditionally independent from T given the current set $\mathbb{MB}$—and so they will be not included to $\mathbb{MB}$. There can be several benefits of such situation. First of all the reliability of the Markov blanket approximation learning in such case can be significantly better—because with the small size of the set $\mathbb{MB}$ we assure the small size of the conditional sets in performed tests as well. Secondly, the execution time of the whole learning process can be reduced, as in the Shrink phase we deal with the smaller set $\mathbb{MB}$, and so we have less variables to process, and smaller conditional relations to consider for each of them.

And of course the enhanced in such way reliability and efficiency of the GS Markov blanket learning subroutine have a direct influence on the reliability and efficiency of the whole GS Bayesian network structure learning methodology, that is Algorithm 4.3.4.1.

In the following chapter we will present in particular an enhancement of the GS algorithm, a significantly more sophisticated solution, for which the motivation was exactly related to applying an additional variables ordering. This enhancement has become the motivation for even more sophisticated solutions, which we will soon see.

Chapter 5

Constraint-Based Learning—Local Discovery Methods

In this chapter we demonstrate the significant development of the constraint-based Bayesian network structure learning methods related to a definitely one of its most fruitful subbranches—the local learning algorithms. The root of this branch and the source of inspiration here is the GS algorithm, which we have described in details in Chapter 4. In particular the main result of this dissertation, the LBNA algorithm presented in Chapter 6, ultimately belongs to the field of local learning algorithms. It is a generalization of the root GS method, with the theoretical background strongly related to the one on which the progress in this branch of algorithms is based.

5.1. Incremental Association Markov Blanket

We start with the algorithm which is a direct descendant of GS, in the sense of the ordering improvements aspects—here however slightly more advanced than in the case of the most straightforward idea coined in Subsection 4.3.10. We describe the algorithm and explain its correctness. We discuss some practical issues like the way of resolving neglected tests for conditional independence. We analyze the performance of the algorithm with respect to all aspects defined in Subsection 4.1.2 and all scenarios from Subsection 4.1.3. We end this section with some explanation why this method is indeed a significant improvement of GS.

5.1.1. Algorithm Description

The IAMB (Incremental Association Markov Blanket) algorithm proposed by Tsamardinos et al. (2003a) is an enhancement of the described in Section 4.3 GS method. More precisely—it is an improvement of the GS Markov blanket learning method, that is Algorithm 4.3.3.1. In particular it can be used as an alternative Markov blanket learning subroutine in the GS Bayesian network learning methodology, that is in Algorithm 4.3.4.1. And, as we will explain it, such replacement is very profitable.

Similarly as the GS method IAMB have on its input some dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ and one chosen target attribute $T \in \mathcal{V}$. As usually we will assume in all further analysis that $\mathcal{D}$ consists of m objects and n attributes. The aim of the method is also the same—it returns one subset of $\mathcal{V}$ corresponding to the learned Markov blanket of T. Moreover, the algorithm has analogical theoretical properties. In the case when for the given on the input sample there exists a generative distribution $\mathcal{P}$ admitting a faithful representation, IAMB returns

the correct solution, corresponding to the true and unique Markov blanket of T with regard to the distribution $\mathcal{P}$—provided that it has ability to recognize always correctly any requested conditional independence.

The IAMB modification of the GS Markov blanket learning approach corresponds exactly to the mentioned at the end of the previous chapter aspects. Namely, some additional variables ordering in the blanket learning mechanism is applied here. The explained there ordering is a computationally light solution. The IAMB ordering of variables is in general in a pessimistic case a bit heavier, and it is not performed a priori but during the blanket learning process. However, as we will see it in this section in the most interesting scenarios this ordering is light as well, and in the same time it is significantly stronger. Some benefits of this strength are worth much interest.

Algorithm 5.1.1.1 is the IAMB algorithm. As we can see at the high level of abstraction it is a very similar method to GS. It also consists of two phases: the Grow phase and Shrink phase, where the analogical set $\mathbb{MB}$ corresponding to the learned blanket is in the first phase growing, while in the second phase shrinking. The Shrink phase is in fact identical to the one applied in GS. The main difference appears in the Grow phase, where the mentioned advanced variables ordering appears.

Algorithm 5.1.1.1 The IAMB Markov blanket learning algorithm.

---

1. Let us initialize the set $\mathbb{MB}$ representing the Markov blanket $MB_T^{\mathcal{P}}$ as an empty set.
2. (Grow phase) Repeat:
   - Choose one variable $X \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}$ for which the value $Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB})$ is maximal. If the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ does not hold add X into the set $\mathbb{MB}$, otherwise terminate the repeat loop.
3. (Shrink phase) One time proceed each variable belonging after the Grow phase to the set $\mathbb{MB}$, in any order. If for the currently considered variable X the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB} \setminus \{X\})$ holds remove X from the set $\mathbb{MB}$.
4. Return the set $\mathbb{MB}$.

This order is in fact very natural. Let us remind that in the case of the Grow phase in the GS algorithm each next processed variable X was included to $\mathbb{MB}$ if the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ does not hold. Exactly the same situation is here, however firstly there is performed the choice of such next processed variable X—and this choice is done with respect to exactly the same requested conditional dependence relation. We simply choose as the next processed variable X this one, which is maximally dependent with T given the current set of variables $\mathbb{MB}$. Our measure of dependence is here the mentioned at the end of Chapter 4 and introduced in Subsection 2.2.6 in Equation 2.2.6.1 $Assoc_{\mathcal{P}}$ measure.

We have in fact explained the motivation standing behind the variables ordering in the GS mechanism. Briefly telling, we want to place all variables from the Markov blanket of T in $\mathbb{MB}$ as early as possible, thanks to which the more reliable and efficient is the whole learning method. Here this potential efficiency improvement is even more visible—in the modified construction of the algorithm itself. Namely, notice that there are two more differences in the Grow phase of IAMB comparing to GS. First of all, as we can see only one run through all the processed variables is performed in the Grow phase. In GS two runs were necessary. Secondly, in the case when the next processed variable X satisfies $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ not only

X is not inserted to $\mathbb{MB}$ as in the case of GS, but also the whole process of processing any further variables is aborted. Any unprocessed variable will be for sure not included to $\mathbb{MB}$ in such case. The theoretical soundness of such only one run and its potential abortion is guaranteed exactly thanks to the strong variables ordering.

5.1.2. Correctness Analysis

The important requirement for the mentioned theoretical soundness of the IAMB algorithm is the consistency of the applied measure $Assoc_{\mathcal{P}}$ and performed statistical tests for conditional independence resolving any requested conditional independence relation $Ind_{\mathcal{P}}$. Namely, in order to assure this soundness the measure $Assoc_{\mathcal{P}}$ must be in the form given in Equation 2.2.6.1, that is for any different variables $X, Y \in \mathcal{V}$ and a subset $\mathbb{Z} \subseteq \mathcal{V} \setminus \{X, Y\}$ it must satisfy $Assoc_{\mathcal{P}}(X, Y \mid \mathbb{Z}) = 1 - p$ where p is the p-value of the performed statistical test resolving the corresponding $Ind_{\mathcal{P}}(X, Y \mid \mathbb{Z})$ relation.

Given the above assumption, together with the assumption of the existence of the generative distribution $\mathcal{P}$ admitting a faithful representation and the perfect conditional independence recognition provided by the tests for conditional independence, the soundness of IAMB can be derived as follows.

What we only need to check is that after the Grow phase there holds $MB_T^{\mathcal{P}} \subseteq \mathbb{MB}$. The reason is that the Shrink phase remains in IAMB the same as in the case of GS, and so the property $MB_T^{\mathcal{P}} = \mathbb{MB}$ occurring after performing this phase we have already proved in Subsection 4.3.3.

Let us consider the moment in the repeat loop in the Grow phase of Algorithm 5.1.1.1 when we choose as a next processed variable X maximizing the value $Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB})$ such one that the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ holds (if such moment does not occur, then we simply have the situation when we gather all variables to $\mathbb{MB}$ in the Grow phase, so obviously we have that $MB_T^{\mathcal{P}} \subseteq \mathbb{MB}$). This conditional independence means in particular that the p-value of the performed statistical test investigating it was above the significance level. Such p-value is equal to $1 - Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB})$. Moreover, for any other still unprocessed variable $Y \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}$ there holds that the p-value of the test investigating the conditional independence $Ind_{\mathcal{P}}(Y, T \mid \mathbb{MB})$ is equal to $1 - Assoc_{\mathcal{P}}(Y, T \mid \mathbb{MB})$. For the chosen variable X the value $Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB})$ is maximal among all still unprocessed variables—so the p-value of the test investigating the corresponding to it conditional independence is the smallest. This in turn means that for all $Y \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}$ the p-value of the test investigating the conditional independence $Ind_{\mathcal{P}}(Y, T \mid \mathbb{MB})$ is above the significance level, so for all such variables Y the conditional independence $Ind_{\mathcal{P}}(Y, T \mid \mathbb{MB})$ would be clearly returned by these tests. According to our theoretical assumptions the tests are providing us a perfect knowledge about independencies, so we have that for each $Y \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}$ the conditional independence $Ind_{\mathcal{P}}(Y, T \mid \mathbb{MB})$ holds. Again according to the assumptions $\mathcal{P}$ admits a faithful representation, so there exists a DAG $\mathcal{G}$ which is a perfect map of $\mathcal{P}$, in which all d-separations correspond to conditional independencies occurring in $\mathcal{P}$. In particular for all $Y \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}$ there appears in $\mathcal{G}$ the d-separation $Ind_{\mathcal{G}}(Y, T \mid \mathbb{MB})$. Directly from the definition of d-separation (see Definition 2.3.4.3) we obtain then that in $\mathcal{G}$ there exists also the d-separation $Ind_{\mathcal{G}}(\mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}, T \mid \mathbb{MB})$. $\mathcal{G}$ is a perfect map of $\mathcal{P}$, so the conditional independence $Ind_{\mathcal{P}}(\mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}, T \mid \mathbb{MB})$ also holds. The Markov blanket of T with regard to $\mathcal{P}$ is unique according to Theorem 4.3.1.1, so the induced conditional independence simply means that $MB_T^{\mathcal{P}} \subseteq \mathbb{MB}$.

5.1.3. Dealing with Neglected Tests

Let us say few words about applying here the Spirtes et al. (2000) convention of neglecting the least reliable tests for conditional independence. As we remember the tests for which the size of the conditional set in the investigated relation exceeds some value (at most $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$ in the case of binary attributes, where c is a defined by a user constant) are neglected, and either the conditional independence or conditional dependence is assumed. In the case of IAMB the situation is very similar to the case of GS. Namely, the automatic assumption of conditional dependence in the case of neglected tests would be dangerous, as it could easily lead to the production of very big supersets of Markov blankets consisting of all in fact variables. Whenever the set $\mathbb{MB}$ exceeds the size b_m in the Grow phase for all further processed variables the corresponding conditional dependence will be automatically returned, so all of them will be placed in $\mathbb{MB}$—and of course in the Shrink phase they will not be removed as all appearing there tests will be neglected.

So in practice the same convention as for the GS method is here applied—for each neglected test the conditional independence is automatically assumed. Similarly as in GS in such setting the Markov blanket returned by IAMB cannot exceed size $b_m + 1$, as when the size of the set $\mathbb{MB}$ exceeds b_m the first further test for conditional independence automatically returns an independence and the Grow phase is aborted. As a consequence, when we use IAMB as a Markov blanket learning subroutine in the GS Bayesian network structure learning algorithm the obtained Markov blankets of each variable does not exceed this size. So similarly as in the case of GS in IAMB there is no risk with automatic conditional independence assumption in the context of further network learning—the risk which in the case of the algorithms SGS and PC could lead to the total degeneration of the learned structure and obtaining an empty graph. Namely, thanks to the limited Markov blankets no conditional independence tests will be neglected during the skeleton construction, that is in Step 3 of Algorithm 4.3.4.1.

5.1.4. Performance—Comparison Scheme

Let us analyze the IAMB algorithm in similar fashion as the other constraint-based algorithms in Chapter 4—with regard to four introduced in Section 4.1 scenarios and three aspects. We will analyze the performance of IAMB with respect to its two applications:

- Markov blanket learning: in this case we compare IAMB with corresponding to it GS Markov blanket learning algorithm, that is with Algorithm 4.3.3.1.

- Bayesian network structure learning: here we assume that we apply IAMB in the role of the auxiliary Markov blanket learning subroutine in the GS Bayesian network structure learning algorithm, that is in Algorithm 4.3.4.1.

Whichever application we consider the only required thing to do here is to analyze the performance of the IAMB algorithm itself. In the case of its application in the GS network structure learning method the performance will be directly derived from the IAMB performance and already analyzed performance of Steps 3 and 5 of Algorithm 4.3.4.1. The reason is that both GS and IAMB Markov blanket learning methods have the same pessimistic size of the resultant blanket in each of the four scenarios, while the performance of the mentioned Steps 3 and 5 depends exactly on this size.

For the purpose of analyzing the second application let us simplify the notation a bit. Namely let us denote by:

- GS-GS the Bayesian network structure learning method resulting from applying in Algorithm 4.3.4.1 (the GS Bayesian network structure learning method) an auxiliary GS Markov blanket learning method.
- GS-IAMB the Bayesian network structure learning method resulting from applying in Algorithm 4.3.4.1 an auxiliary IAMB Markov blanket learning method.

5.1.5. Performance—Scenario ST_1 and SP_1

As usually we start with the theoretical scenario ST_1. Let us first consider the pessimistic number of performed statistical tests for conditional independence:

- In the Grow phase of Algorithm 5.1.1.1 we pessimistically can include all variables to $\mathbb{MB}$ without any earlier termination, so we pessimistically perform here $\mathcal{O}(n^2)$ tests—as in order to add each of the $\mathcal{O}(n)$ elements to $\mathbb{MB}$ we perform in fact $\mathcal{O}(n)$ tests in order to obtain the $Assoc_{\mathcal{P}}$ values.
- In the shrink phase, similarly as in the case of GS, we pessimistically perform $\mathcal{O}(n)$ tests.

So totally we perform $\mathcal{O}(n^2)$ tests. The time complexity of Algorithm 5.1.1.1 can be summarized as follows:

- In the Grow phase we perform $\mathcal{O}(n^2)$ tests. Each performed test costs pessimistically $\mathcal{O}(mn)$ time—but similarly as in the case of GS it is possible to appropriately store and efficiently update for each next iteration of the repeat loop the conditional frequencies used in the calculated test statistics, and as the result the appropriately implemented Grow phase costs $\mathcal{O}(mn^2)$ time.
- In the Shrink phase we perform only $\mathcal{O}(n)$ tests, but there are no possible such implementation improvements in the calculation of test statistics as in the Grow phase. As the result, the Shrink phase costs also $\mathcal{O}(mn^2)$ time.

The overall time complexity of the IAMB algorithm is $\mathcal{O}(mn^2)$. What remains to analyze is the pessimistic size of the conditional set in performed tests. But clearly this size is equal to $n-2$.

Summarizing, we can see that although in the IAMB algorithm we perform $\mathcal{O}(n^2)$ tests, while in the GS Markov blanket learning method only $\mathcal{O}(n)$, the time complexity of both methods is the same: $\mathcal{O}(mn^2)$. The pessimistic size of the conditional set in performed tests is in case of both methods equal to $n-2$.

Both methods are theoretically sound, so when we apply them as an auxiliary Markov blanket learning tool in Algorithm 4.3.4.1 their behavior there is equivalent—they both return exactly the same information about all Markov blankets, as both these algorithms return in the considered here theoretical scenario optimal exact solutions. So the performance of Steps 3 and 5 of Algorithm 4.3.4.1 will be the same in the case of applying both of these Markov blanket learning methods—we exactly know this performance, as we have analyzed it in Subsection 4.3.7.

According to what we have found there we can now conclude that in the aspects of the time complexity and pessimistic size of the conditional set in performed tests all four algorithms: SGS, PC, GS-GS and GS-IAMB are equal in the scenario ST_1—they have the time complexity exponential with regard to n and linear with regard to m, and their pessimistic size of the conditional set is $n-2$.

Although in the case of GS-IAMB Step 1 of Algorithm 4.3.4.1 requires performing $\mathcal{O}(n^3)$ tests, while in the case of GS-GS this step requires only $\mathcal{O}(n^2)$ tests—this difference is negligible, as the remaining steps in both algorithms GS-GS and GS-IAMB require pessimistically exponential with regard to n number of performed tests. As the result we obtain that all four so far considered Bayesian network structure learning algorithms requires pessimistically performing exponential with regard to n number of tests for conditional independence.

The first practical scenario SP_1 turns out to be also in the case of IAMB indistinguishable from the scenario ST_1. The performance of the IAMB Markov blanket learning method as well as the performance of the network structure learning algorithm GS-IAMB remains—similarly as the performance of the GS Markov blanket learning Method, GS-GS, SGS and PC—the same as in the scenario ST_1. Although the learned Markov blankets in the case of GS and IAMB are not here guaranteed to be the same, in the case of lack of any assumptions, which is exactly the nature of the scenario SP_1, we can derive the same in both cases conclusions about the pessimistic size of these blankets—which is the same as in the scenario ST_1, that is $n - 1$.

5.1.6. Performance—Scenario ST_2

Let us now consider the second theoretical scenario ST_2, where in the skeleton characterizing any faithful representation of the generative distribution each node has degree at most k. As we remember from Subsection 4.3.8 the size of the (unique) Markov blanket of any variable is in such case limited by k^2.

The number of performed tests can be in this case summarized as follows: nothing changes comparing to the scenario ST_1. The order of variables might be helpful in practice, but it does not guarantee any earlier termination of the repeat loop in the Grow phase in Algorithm 5.1.1.1. So the pessimistic number of performed tests in IAMB is here also $\mathcal{O}(n^2)$.

Consequently nothing also changes with respect to the time complexity and pessimistic size of the conditional set: they are respectively for IAMB equal to $\mathcal{O}(mn^2)$ and $n - 2$. So in the case of the scenario ST_2 both GS and IAMB Markov blanket learning algorithms have the same time complexity: $\mathcal{O}(mn^2)$ and the same pessimistic size of the conditional set: $n - 2$—although GS requires performing $\mathcal{O}(n)$ tests, while IAMB $\mathcal{O}(n^2)$ tests.

As the result, both GS-GS and GS-IAMB algorithms are equally strong with respect to the execution time: $\mathcal{O}(mn^3)$, and the pessimistic size of the conditional set: $n - 2$. GS-GS requires however performing only $\mathcal{O}(n^2)$ tests, while GS-IAMB $\mathcal{O}(n^3)$ tests. Both methods are with respect to the efficiency much better in the scenario ST_2 than the SGS and PC algorithms—although we must remember that PC is a leader in this scenario with respect to the reliability, with the maximal size of the conditional set in performed tests equal to k.

5.1.7. Performance—Scenario SP_2

The second practical scenario SP_2 requires definitely more attention. It corresponds to the exactly the same setting as the one considered for algorithm GS, where whenever the size of the conditional set in the performed test for conditional independence exceeds value $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$ the test is aborted and the corresponding conditional independence is automatically returned.

Let us first consider the pessimistic number of performed tests:

- In the Grow phase the repeat loop is for sure instantly aborted whenever the size of the set $\mathbb{MB}$ exceeds value b_m—as in such case whichever remaining variable X will be processed, the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB})$ will be automatically

assumed. In each repetition of this loop one variable is added to $\mathbb{MB}$, so we clearly obtain that there can be no more than $b_m + 1$ such repetitions (in the eventual next repetition we can automatically terminate the whole loop). In each repetition we perform $\mathcal{O}(n)$ tests in order to calculate the $Assoc_{\mathcal{P}}$ values, so totally in the Grow phase we perform $\mathcal{O}(b_m n)$ tests, that is $\mathcal{O}((\log_2 m)n)$ tests.

- Consequently in the Shrink phase we perform $\mathcal{O}(b_m)$ tests, as the size of the set $\mathbb{MB}$ after the Grow phase is bounded by $b_m + 1$.

So totally in IAMB we pessimistically perform here $\mathcal{O}((\log_2 m)n)$ tests for conditional independence. The time complexity of IAMB is following:

- In the Grow phase we perform $\mathcal{O}(b_m n)$ tests, where each of them costs pessimistically $\mathcal{O}(mb_m)$ time (the conditional set does not exceed size b_m). But efficient implementation with structures storing auxiliary frequencies updated at each next repetition of the loop enables performing each test in $\mathcal{O}(m)$ time. As a consequence the time complexity of the Grow phase is $\mathcal{O}(mb_m n)$, that is $\mathcal{O}(m(\log_2 m)n)$.

- In the Shrink phase we perform $\mathcal{O}(b_m)$ tests, where each pessimistically costs $\mathcal{O}(mb_m)$ time. No implementation improvements are available here, so the time complexity of this phase is $\mathcal{O}(mb_m^2)$ that is $\mathcal{O}(m(\log_2 m)^2)$.

The overall IAMB time complexity is $\mathcal{O}(m(\log_2 m)n + m(\log_2 m)^2)$. Note that this complexity is in fact equal to $\mathcal{O}(m(\log_2 m)n)$—as whenever there holds the inequality $b_m \geqslant n$ in all above considerations the pessimistic counter b_m can be replaced with $n-2$, because no larger conditional sets in performed statistical tests for conditional independence can appear.

Finally we can immediately notice that the pessimistic size of the conditional set in performed statistical tests for conditional independence is here b_m.

Summarizing, as we can see both GS and IAMB Markov blanket learning methods are similarly reliable in the case of the scenario SP_2. Their pessimistic size of the conditional set is in the case of both of them equal to b_m. The GS method requires performing only $\mathcal{O}(n)$ tests, while the IAMB approach requires more—$\mathcal{O}((\log_2 m)n)$ tests. However, regardless of this difference, both approaches have the same time complexity—$\mathcal{O}(m(\log_2 m)n)$.

Clearly in the scenario SP_2 both GS and IAMB Markov blanket learning approaches produce Markov blankets with analogical restrictions regarding them—their size is bounded by $b_m + 1$. So the performance of Step 3 and 5 of Algorithm 4.3.4.1 in all aspects is the same regardless of the auxiliary Markov blanket learning method applied in Step 1: either GS or IAMB.

Consequently, we obtain that in the scenario SP_2:

- The pessimistic number of performed tests is slightly different in the case of GS-GS and GS-IAMB algorithms. For GS-GS it is $\mathcal{O}(m(\log_2 m)^2 n + n^2)$, while for GS-IAMB: $\mathcal{O}(m(\log_2 m)^2 n + (log_2 m)n^2)$. Let us recall that the performance of SGS and PC in this aspect is much worse than in the case of GS-GS and GS-IAMB, as although the number of performed tests is there polynomial with respect to both m and n, each of the two exponents directly depends on the second dimension of the table.

- The crucial observation is that the slightly worse GS-IAMB performance comparing to GS-GS in the aspect of the number of performed tests is not visible with respect to the time complexity, where both approaches GS-GS and GS-IAMB achieves the complexity $\mathcal{O}((m \log_2 m)n^2 + m^2(log_2 m)^3 n)$. The complexity of the SGS and PC algorithm in turn is again problematic in the same sense as in the point above.

- The reliability of all so far considered methods: SGS, PC, GS-GS and GS-IAMB is the same—the pessimistic size of the conditional set in performed tests is equal to b_m.

5.1.8. IAMB Ordering Strength

A very clear conclusions can be drawn from the whole presented above analysis of the IAMB algorithm with respect to our four scenarios and three aspects. The most interesting for us is here the comparison of IAMB with its predecessor—the GS Markov blanket learning subroutine. We have reached to a very interesting observations here. Namely, both the GS and IAMB approaches have turned out to be equivalent with respect to pessimistic bounds regarding their performance in both efficiency and reliability aspects. In all four scenarios they both have the same pessimistic bounds regarding the time complexity and the size of the conditional set in performed tests—regardless of the fact that IAMB is performing more tests for conditional independence. Moreover, also after applying these two methods as a subroutine in Algorithm 4.3.4.1—the GS Bayesian network structure learning method, both resultant algorithms GS-GS and GS-IAMB have the same bounds in aforementioned aspects, and in all scenarios.

So what is the real value of IAMB ? Without doubts it is the strong ordering strategy of processed in the Grow phase variables. It can definitely make a difference in practical applications—where the quality of the results provided by IAMB can be in fact much better than in the case of GS. Although this difference is not visible in the worst case settings considered here—the real-live scenarios are not corresponding to them. The IAMB ordering of variables is even stronger than the one simpler heuristic approach proposed by the authors of GS—and it definitely can lead to incomparably smaller growth of the set $\mathbb{MB}$ in the Grow phase, which is a key aspect if we are interested in the reliability of performed statistical test for conditional independence—and as the result the reliability of the whole learning approach.

But these discussed here worst case scenarios show in turn that the IAMB method is able to remain in the worst case equally well scalable as the GS approach, which is its big advantage, if we keep in mind that IAMB provides an advanced and complicated ordering heuristic. The strength of the IAMB ordering heuristic comparing to the plain GS solution is confirmed in practice. An interested reader can look for example at the Tsamardinos et al. (2003a) work, where the empirical evaluation does not leave any doubts in this matter.

An interesting property of the IAMB ordering is that it has some additional theoretical properties, which enables to design the sound Markov blanket learning algorithm relaying only on this ordering—without performing any additional conditional independence tests. A reader interested in this topic can look at the Betliński (2011) work, where such additional IAMB ordering properties are investigated together with the proposed blanket learning method basing on these properties.

5.2. Max-Min Parents and Children

The IAMB method was a great step towards improving especially in practice the performance of the so far having no serious competitors Grow-Shrink algorithm. But what is even more important, IAMB has very soon became an inspiration for the next local learning approach, which has turned out to be a crucial point—the point which has established in fact the paradigm of the local discovery methods within the field of constraint-based Bayesian network structure learning methods. This subsection covers several aspects regarding this approach, including: algorithm description, applying the Spirtes et al. (2000) convention for neglected tests, and the analysis of the performance in all aspects defined in Subsection 4.1.2 and

all scenarios from Subsection 4.1.3. The considered method turns out to be only partially theoretically sound, which we explain in details with an example showing where the lack of correctness lies. Nevertheless as we will see there are several reasons to claim that this approach is a next milestone in the field of constraint-based solutions.

5.2.1. Crucial Idea

In this section we demonstrate a very significant solution proposed by Tsamardinos et al. (2003b)—the MMPC (Max-Min Parents and Children) algorithm—which is from one side an improvement of the IAMB method, but from the other side also a significantly new, successful and fruitful approach.

The crucial observation achieved by the MMPC authors is that the (proposed by themselves also, by the way) IAMB ordering of variables can be even further strengthen—and the whole approach can be transformed into the method which learns a significantly smaller than the Markov blanket local structure around the target node.

Namely, if we assume that for the given on the entry sample $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ there exists a generative distribution $\mathcal{P}$ admitting a faithful representation—the MMPC method is aimed at finding for the given input target attribute—variable T the set of its neighbors (that is parents and children—as the name of the method suggests) in any perfect map of $\mathcal{P}$.

Although separately considered sets of parents and children for a given node T can be different depending on the chosen perfect map of $\mathcal{P}$—the point of interest here, that is the together taken both these sets—in other words the set of neighbors of T, remains the same whatever perfect map we choose. This fact is an immediate consequence of Theorem 2.4.3.1—simply in particular all perfect maps have the same skeleton and the same v-structures. On the same fact was based an analogical property of Markov blankets, which we explained in Subsection 4.3.1 with an assist of Theorems 4.3.1.1 and 4.3.1.2.

Similarly as the Markov blanket set of a given variable T with respect to distribution $\mathcal{P}$ we denote as $MB_T^{\mathcal{P}}$, the set of neighbors of T in the skeleton characterizing the Markov equivalence class of perfect maps of distribution $\mathcal{P}$ we will denote here as $NB_T^{\mathcal{P}}$.

5.2.2. Algorithm Description

Algorithm 5.2.2.1 is the MMPC method. As it can be noticed, its structure is on the high level very similar to the previous GS and IAMB methods. Again we construct the desired set starting from an empty set—here called $\mathbb{NB}$. Again there are two phases here: the Grow phase, in which we potentially add some elements to the set $\mathbb{NB}$, and the Shrink phase, where we potentially remove some of the elements from this set.

The second part of the MMPC method name we have already explained. The first part corresponds to the applied here much stronger than previously variables ordering in the Grow phase—so called Max-Min ordering. This ordering in conjunction with the loop termination rule in the Grow phase and variables removal condition in the Shrink phase is so strong that it would be no longer applicable in the case of searching for the Markov blanket of T. Note that the independence of the currently considered variable X from T given some subset of remaining variables does not necessarily mean that X does not belong to $MB_T^{\mathcal{P}}$. For example, it is possible that $X \in MB_T^{\mathcal{P}}$ and in the same time even the most straightforward independence $Ind_{\mathcal{P}}(X, T \mid \emptyset)$ holds. This can happen when X is not a direct neighbor of T in the skeleton characterizing the class of perfect maps of $\mathcal{P}$, but it is only a parent of some children of T in these perfect maps. However, the MMPC method is searching for such conditional independence evidences - and they are enough to stop the loop in the Grow phase

Algorithm 5.2.2.1 The MMPC neighbors (parents and children) learning algorithm.

1. Let us initialize the set $\mathbb{NB}$ representing the set of neighbors $NB_T^{\mathcal{P}}$ as an empty set.
2. (Grow phase) Repeat:
   - Choose one variable $X \in \mathcal{V} \setminus \{\mathbb{NB} \cup \{T\}\}$ for which the value $MinAssoc_{\mathcal{P}}(X, T \mid \mathbb{NB})$ is maximal, where for any $Y \in \mathcal{V}$:

     $$MinAssoc_{\mathcal{P}}(Y, T \mid \mathbb{NB}) = \min_{\mathbb{S} \subseteq \mathbb{NB}} Assoc_{\mathcal{P}}(Y, T \mid \mathbb{S}).$$

     Let $\mathbb{Z} = \arg\min_{\mathbb{S} \subseteq \mathbb{NB}} Assoc_{\mathcal{P}}(X, T \mid \mathbb{S})$. If the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{Z})$ does not hold add X into the set $\mathbb{NB}$, otherwise terminate the repeat loop.
3. (Shrink phase) One time proceed each variable belonging after the Grow phase to the set $\mathbb{NB}$, in any order. If for the currently considered variable X the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{S})$ holds for some $\mathbb{S} \subseteq \mathbb{NB} \setminus \{X\}$ remove X from the set $\mathbb{NB}$.
4. Return the set $\mathbb{NB}$.

and so eliminate all not inserted so far variables or eliminate any processed variable in the Shrink phase. The crucial property which we will soon see is that this strategy does not eliminate any neighbors of T.

5.2.3. Wise Aggressiveness

MMPC is a far more aggressive method than IAMB—which in practice leads to significantly higher than IAMB reliability of performed learning. MMPC typically deals with conditional sets appearing in conducted statistical tests for conditional independence of incomparably smaller size than in IAMB, where the size of approximated set $\mathbb{MB}$ tends to grow much more than the approximated set $\mathbb{NB}$ in MMPC. This whole aggressiveness of MMPC appears in both the Grow and Shrink phase, and it can be summarized as follows:

- The most standard convention is when any employed in the IAMB and MMPC method measure of association $Assoc_{\mathcal{P}}(X, T \mid \mathbb{S})$ is calculated as $1 - p$ where p is the p-value of the statistical test for conditional independence investigating the relation $Ind_{\mathcal{P}}(X, T \mid \mathbb{S})$, and the same test is used in order to resolve any conditional independence relation and potentially stop the repeat loop in the Grow phase. The strength of MMPC in the Grow phase can be in such case expressed as follows. Imagine that after l repetitions of the repeat loop in the Grow phase of IAMB and after l repetitions of the repeat loop in the Grow phase of MMPC applied on the same input dataset and for the same target variable T there holds $\mathbb{MB} = \mathbb{NB}$. In such case we have that in the next repetition of the loop the following inequality for any remaining not inserted to the growing set variable X holds:

  $$MinAssoc_{\mathcal{P}}(X, T \mid \mathbb{NB}) \leqslant Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB}),$$

  as in general for any $X, Y \in \mathcal{V}$ and $\mathbb{S} \subseteq \mathcal{V} \setminus \{X, Y\}$ there always holds an obvious

inequality:

$$MinAssoc_{\mathcal{P}}(X, Y \mid \mathbb{S}) \leqslant Assoc_{\mathcal{P}}(X, Y \mid \mathbb{S}).$$

As a consequence, there also holds:

$$\max_{X \in \mathcal{V} \setminus \{\mathbb{NB} \cup \{T\}\}} MinAssoc_{\mathcal{P}}(X, T \mid \mathbb{NB}) \leqslant \max_{X \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}} Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB}),$$

and so:

$$1 - \max_{X \in \mathcal{V} \setminus \{\mathbb{NB} \cup \{T\}\}} MinAssoc_{\mathcal{P}}(X, T \mid \mathbb{NB}) \geqslant 1 - \max_{X \in \mathcal{V} \setminus \{\mathbb{MB} \cup \{T\}\}} Assoc_{\mathcal{P}}(X, T \mid \mathbb{MB}).$$

The compared here values corresponds exactly to the p-values of conditional independence tests which are deciding whether to stop the repeat loop in the Grow phase of respectively MMPC (left hand side) and IAMB (right hand side). The bigger p-value in the case of MMPC simply means that whenever in such situation the IAMB Grow phase is aborted the MMPC Grow phase is aborted too. In other words the MMPC Grow phase is stopped no later than the IAMB Grow phase.

- In the Shrink phase of the IAMB algorithm we remove any variable X from $\mathbb{MB}$ only when the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{MB} \setminus \{X\})$ holds. In the MMPC algorithm each processed variable X in the Shrink phase is removed from $\mathbb{NB}$ whenever even a one conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{S})$ for any $\mathbb{S} \subseteq \mathbb{NB} \setminus \{X\}$ holds. This in practice can lead to the much more frequent removal of the variables, and a faster reduction of the $\mathbb{NB}$ set.

The MMPC method is indeed very aggressive, but in the same time it can be easily noticed that this aggressiveness leads to an exhaustive search of appropriate subsets of variables, and so potentially a very time consuming learning. However, it is also important to notice that this exhaustive search is conducted only withing the current set $\mathbb{NB}$.

One could imagine the hypothetical algorithm, where each search in order to calculate the $MinAssoc_{\mathcal{P}}$ value is performed among all possible subsets of remaining variables. In such case the Grow phase would be enough, as it is easy to see that it would be already the theoretically correct solution returning the desired neighborhood. Namely, for all variables not belonging to the desired neighborhood according to Remark 2.4.4.1 the conditional independence evidence would be found, and for all variables from the neighborhood such evidence cannot be found, and so—assuming the perfect behavior of statistical tests—the $MinAssoc_{\mathcal{P}}$ value corresponding to any variable outside of the neighborhood would be smaller than for any variable belonging to the neighborhood. Consequently, all the neighbors would be consecutively included in the Grow phase and for the first variable outside the conditional independence evidence results in terminating the loop. The only problem of such solution however is that its practical usefulness would be dimmed.

So the solution provided in the MMPC method is a kind of trade off between the efficiency of the approach and the reduction of the growth of the created set $\mathbb{NB}$. We focus here on searching only for the conditional subsets included in the current set $\mathbb{NB}$. The motivation standing behind such choice corresponds directly to Remark 2.4.4.3. The consequence of this remark is that if two variables X and Y are not connected in the perfect map of the joint probability distribution then they are conditionally independent given at least one of the sets: either the set of neighbors of X or the set of neighbors of Y in such perfect map. What we are trying to catch is exactly such set. The set $\mathbb{NB}$ grows during the Grow phase and in particular it should contain after some number of repetitions most of the variables from

the neighborhood of the target variable T. So for each next processed variable in the Grow phase there is an increasing chance that we catch the subset corresponding to the conditional independence evidence—whenever it can be found in the desired neighborhood of T.

5.2.4. Correctness Analysis

If we assume that there exists a generative distribution of the input sample admitting a faithful representation, as well as that the $Assoc_{\mathcal{P}}$ measures are calculated with an assist of the overall used in the method statistical tests for conditional independence, and finally that the results of these tests are always correct, the MMPC method turns out to be only partially sound. This partial soundness corresponds exactly to the aforementioned trade off between efficiency and accuracy in this method. Namely, what can be only shown is that the resultant set $\mathbb{NB}$ representing the learned neighborhood of the input target variable T satisfies $\mathbb{NB} \supseteq NB_T^{\mathcal{P}}$:

- In the Grow phase, if for some currently chosen according to the Max-Min ordering variable X in the repeat loop there holds the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{Z})$ where $\mathbb{Z} = \arg\min_{\mathbb{S} \subseteq \mathbb{NB}} Assoc_{\mathcal{P}}(X, T \mid \mathbb{S})$, then in particular the $MinAssoc_{\mathcal{P}}(X, T \mid \mathbb{NB})$ is sufficiently small and the corresponding p-value of the conditional independence test sufficiently big in order to indicate this conditional independence. The $MinAssoc_{\mathcal{P}}(Y, T \mid \mathbb{NB})$ for all remaining variables $Y \in \mathcal{V} \setminus \{\mathbb{NB} \cup \{X, T\}\}$ is not bigger, so the corresponding p-values expressing the most independent conditional relations regarding them are not smaller—which means that in the case of these relations the conditional independence will be also returned by the tests—and so according to the assumptions these independencies exists. So, according to the mentioned Remark 2.4.4.1, X and all the remaining variables not included so far to $\mathbb{NB}$ for sure does not belong to the desired neighborhood $NB_T^{\mathcal{P}}$.

- In the Shrink phase we exclude any variable X from $\mathbb{NB}$ only if the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{S})$ has been found for some subset $\mathbb{S}$—and in such situation, again according to Remark 2.4.4.1, X for sure does not belong to $NB_T^{\mathcal{P}}$.

Unfortunately, in general there might appear the situation in the considered above theoretical setting where the learned with the assist of MMPC set $\mathbb{NB}$ is a strict superset of the desired set $NB_T^{\mathcal{P}}$. Let us first give some intuition why this can happen.

Let us consider some variable X belonging after the Grow phase to the set $\mathbb{NB} \setminus NB_T^{\mathcal{P}}$. According to aforementioned earlier Remark 2.4.4.3 there holds for sure the conditional independence $Ind_{\mathcal{P}}(X, T \mid \mathbb{S})$ for either some subset $\mathbb{S} \subseteq NB_T^{\mathcal{P}}$ or some subset $\mathbb{S} \subseteq NB_X^{\mathcal{P}}$. There must appear such independence for some subset of at least one of the sets $NB_T^{\mathcal{P}}$ and $NB_X^{\mathcal{P}}$—but not necessarily for both of them. In particular if we are unlucky and the conditional independence evidence cannot be found for subsets included in $NB_T^{\mathcal{P}}$, then it is possible that the variable X will be not eliminated from $\mathbb{NB}$ in the Shrink phase.

5.2.5. Theoretical Weak Spot—Example

Let us now illustrate this problem very precisely with an example, taken from the Tsamardinos et al. (2006) work. Namely, let us assume that the generative distribution $\mathcal{P}$ of the given on the input of MMPC sample admits a faithful representation, in the form of the DAG presented in Figure 5.2.5.1 (such distribution exists, which we explain later in Subsection 6.2.4, namely with Theorem 6.2.4.2). Our aim is to determine the set $NB_T^{\mathcal{P}}$ for some given on the input

chosen variable T. Let us imagine that we apply the MMPC approach in order to achieve this.

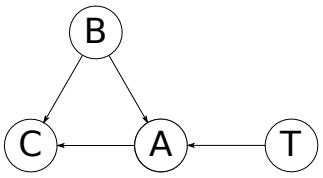


Figure 5.2.5.1: Picture of an exemplary faithful representation of the generative distribution of the input sample, for which the lack of soundness of the MMPC approach is visible.

Assuming that the considered in the proof of the partial MMPC soundness theoretical settings hold, we have that:

- In the Grow phase we will for sure include the variable A to the set $\mathbb{NB}$—because (according to Remark 2.4.4.1) it is dependent with T given any subset of remaining variables, so in each repetition of the loop the calculated $MinAssoc_{\mathcal{P}}$ value corresponding to it will always indicate the conditional dependence.

  We will for sure not include B to $\mathbb{NB}$, because it is independent from T given the empty set—which means that the corresponding to the B variable $MinAssoc_{\mathcal{P}}$ value will indicate the independence at each repetition of the loop (the independence of B and T is a consequence of their d-separation by the empty set in the perfect map of $\mathcal{P}$).

  As a consequence we will for sure include the variable C to $\mathbb{NB}$. What we already know is that B will not be included to $\mathbb{NB}$, so let us consider two possible cases regarding the remaining variables: A and C. In the case when the first included to $\mathbb{NB}$ variable is C—then we are done. Otherwise the first included to $\mathbb{NB}$ variable is A and after this operation the next included variable must be C. The reason is that in the case of each two possible conditional subsets $\mathbb{S}$ included in the current $\mathbb{NB}$: either $\emptyset$ or $\{A\}$, the conditional independence $Ind_{\mathcal{P}}(C, T \mid \mathbb{S})$ clearly does not hold—as the corresponding d-separations in the perfect map of $\mathcal{P}$ does not hold (for the case of $\mathbb{S} = \emptyset$ the chain $[C, A, T]$ is not blocked, while for the case of $\mathbb{S} = \{A\}$ the chain $[C, B, A, T]$ is not blocked).

  Summarizing, after the Grow phase we have that $\mathbb{NB} = \{A, C\}$.

- In the Shrink phase the variable A will be not excluded from $\mathbb{NB}$—as we have noticed it in the previous point it is dependent with T given any subset of remaining variables. And what is the key observation here, the variable C will also not be removed from $\mathbb{NB}$—as C is dependent with T given both $\emptyset$ and $\{A\}$, which we already know from the previous point.

  So after the whole MMPC procedure we remain with the set $\mathbb{NB} = \{A, C\}$, while the real neighborhood of T is equal to $\{A\}$.

The source of the appearing here problem is exactly the aspect mentioned earlier. We deal here with the situation where the conditional independence $Ind_{\mathcal{P}}(C, T \mid \mathbb{S})$ does not hold for any subset $\mathbb{S} \subseteq NB_T^{\mathcal{P}} (= \{A\})$, but only for the subset $\mathbb{S} \subseteq NB_C^{\mathcal{P}} (= \{A, B\})$. Namely, the conditional independence $Ind_{\mathcal{P}}(C, T \mid \{A, B\})$ holds. What is only guaranteed in general is that MMPC gathers in the theoretical scenario all required variables— but there is always some risk that it will also include some more unnecessary ones.

Regardless of this lack of full theoretical soundness, the MMPC approach is a very notable and valuable solution, basing on the great idea of changing the aim of the preceding methods like GS and IAMB and learn a significantly smaller local structure much more aggressively than previously, basing on the much more reliable statistical tests—which is a most desired property in the field of constraint-based learning algorithms. Although the method has only partial theoretical correctness, in practice it can result in obtaining significantly more accurate information about the neighborhood of each variable in the faithful representation than the number of other Bayesian network structure learning methods, including previously considered GS-GS and GS-IAMB approaches. The crucial advantage of MMPC lies in its extremely high comparing to other solutions reliability of performed statistical tests—resulting from the much effort here towards reducing the growth of the desired set $\mathbb{NB}$.

Moreover, the complete soundness of the MMPC has been soon achieved—in the modification proposed by Tsamardinos et al. (2006), which will be the topic of the next section, namely of Subsection 5.3.1.

5.2.6. Dealing with Neglected Tests

Similarly as in the case of the GS and IAMB Markov blanket learning algorithms, in the case of the neighborhood learning MMPC algorithm the Spirtes et al. (2000) convention is applied with an automatic assumption of the conditional independence in the case of neglected statistical tests, that is tests corresponding to conditional sets of appropriately big size (where the threshold for this size is at most $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$—in the case of binary attributes) . The reason is analogical as previously—we want to prevent the Grow phase from some abnormal growth. Here such risk is without doubts less possible, as for each processed variable we try to find the conditional independence evidence not for one conditional set corresponding to the whole current set $\mathbb{NB}$—as it was in the case of GS and IAMB, but for all subsets of $\mathbb{NB}$. However, in the case when in the generative distribution of the sample it is hard to find the conditional independence evidence with the conditional set of the small size between any two variables unconnected in the faithful representation of this distribution—we can face the problem where in the Grow phase we indeed include almost all variables.

Note that in the case when the automatic assumption for neglected tests is the conditional independence we prevent very precisely the growth of the set $\mathbb{NB}$ in the Grow phase—similarly as the growth of the set $\mathbb{MB}$ is prevented in the case of the methods GS and IAMB. Namely, the size of $\mathbb{NB}$ cannot exceed value $b_m + 1$, as whenever the set $\mathbb{NB}$ exceeds in the Grow phase the size b_m, for each remaining not included so far to $\mathbb{NB}$ variable we clearly find the conditional independence evidence with the conditional set corresponding to the whole set $\mathbb{NB}$, for which the automatic verdict is this conditional independence.

5.2.7. Applications—Markov Blanket Learning and Feature Selection

Applying the MMPC method for each attribute in the given dataset results in obtaining some approximation of the skeleton characterizing the class of perfect maps of the generative distribution of the sample. There are several ways to use this result, or some part of it.

In particular MMPC can be easily extended to the Markov blanket learning algorithm. In the Tsamardinos et al. (2003b) work such solution has been in fact proposed, together with the MMPC method. It is called MMMB (Max-Min Markov Blanket). The idea is very simple and in the same time very fruitful. Using the MMPC for the target node T, and then for each node included in the learned neighborhood of T, we easily obtain the set of

all variables which are in distance at most 2 to T in the skeleton characterizing the faithful representation of the generative distribution. All variables which are the direct neighbors of T are automatically included to the approximation of the Markov blanket of T, while all remaining variables (in distance 2 to T) are additionally tested. The test for each such variable X is aimed at checking whether there exists a v-structure (X, Z, T) for the variable Z belonging to the neighborhood of T from which the variable X at distance 2 has been found. The verification is done via performing one statistical test for conditional independence—and what is important, the conditional set in this test exceeds at most by 1 the size of maximal conditional sets appearing in the MMPC runs.

We do not go into more details here, as the Markov blanket approximation is not the main topic of this dissertation. An interested reader can look for example to the mentioned Tsamardinos et al. (2003b) work. In particular, it is worth to look at the experiments results, where the superiority of the MMMB among its predecessors, including described here GS and IAMB, in the accuracy of prediction of the Markov blankets, is proven.

Let us only mention that the MMMB solution achieves similar pessimistic performance bounds as the GS and IAMB especially in the most important practical scenario SP_2, which is its great property if we keep in mind the outstanding accuracy of this method. This performance can be in fact easily derived from the performance of the MMPC mechanism—which is discussed in details in the remaining part of this section.

On the basis of the MMPC and MMMB mechanisms there has raised the whole family of the local learning approaches aimed at the Markov blanket learning task.

The most notable modification of the MMPC method is the HITON-PC algorithm (Aliferis et al., 2003). The main difference lies in the modified and simpler in calculation comparing to Max-Min heuristic ordering of variables, and in the introduction of interleaving Grow and Shrink phases, potentially leading to the situation where even the smaller neighborhood supersets $\mathbb{NB}$ are processed. Similarly as the MMPC method, HITON-PC applied to all variables results in obtaining a network skeleton and can be used as a preprocessing step for the Markov blanket induction—see the HITON-MB method described in this paper. Supporting HITON-MB with the wrapper techniques results in one of the most recognizable and successful feature selection algorithm aimed at the classification task, called HITON. A reader interested in the topic of the feature selection algorithms inspired by the notion of Markov blankets is strongly encouraged to study the Aliferis et al. (2010a) and Aliferis et al. (2010b) great overview of such algorithms, summarizing the big effort of the authors in this matter throughout several years, leading in particular to such sophisticated solutions like IAMB, MMMB, or HITON.

5.2.8. Applications—Bayesian Network Structure Learning

An another application of the MMPC algorithm can be related to the Bayesian network structure learning, similarly as it was in the case of GS and IAMB Markov blanket learning algorithms. Note however that the most straightforward idea here, that is applying the algorithm MMMB as a Markov blanket learning subroutine in the GS network structure learning mechanism, that is using it in Step 1 of Algorithm 4.3.4.1, would be a kind of walk around. In Step 3 of GS mechanism we learn the skeleton characterizing the desired class of perfect maps, while this skeleton we would in fact obtain in Step 1 if the MMMB was applied here—simply because MMMB calculated for each variable in particular invokes MMPC for each variable.

Thus rather a direct usage of MMPC and extending it to a Bayesian network structure learning algorithm could be more interesting. One of the possibilities is to utilize the mechanism of v-structures detection around the target node included in the MMMB method.

Namely, after applying MMPC for each variable in order to detect the skeleton, we can perform analogical tests as in MMMB to detect v-structures in the neighborhood of each variable (in the subgraph corresponding to the nodes at distance at most 2), and so detecting all the v-structures characterizing the faithful representation.

There are also another interesting possibilities. In particular, without doubts the most notable application of the MMPC mechanism in the context of Bayesian network structure learning is the MMHC algorithm (Tsamardinos et al., 2006), which will be the main topic of the next section.

5.2.9. Performance—Comparison Scheme

We finish this section with the analysis of the MMPC algorithm with respect to all aspects and scenarios stated in Section 4.1—as usually with regard to the size of the input dataset, that is the number of objects m and number of attributes n. Remember that we are talking here about the most pessimistic cases regarding this performance only. Similarly as in the case of IAMB, we assume that the calculation of the $Assoc_{\mathcal{P}}$ measure is done via the everywhere else used in the method statistical test for conditional independence, that is by means of Equation 2.2.6.1.

In order to compare this performance with the preceding described in this work Bayesian network structure learning algorithms: SGS, PC, GS-GS and GS-IAMB, we will consider the truncated versions of these methods, with the aim of finding the skeleton only—and so without the calculation of v-structures. We will see how an alternative method of obtaining such skeleton, that is applying the MMPC algorithm for each variable, can be placed with regard to the performance in these previously considered skeleton learning mechanisms.

5.2.10. Performance—Scenario ST_1 and SP_1

We start with the theoretical scenario ST_1. Clearly the pessimistic number of performed tests for conditional independence is in this scenario exponential with regard to n. The size of the set $\mathbb{NB}$ can grow to an arbitrary large in the Grow phase, in particular to the one containing all remaining variables, while during the growth of this set we perform tests for conditional sets corresponding to all subsets of $\mathbb{NB}$. As a consequence the time complexity of the approach is exponential with regard to n too, and as usually it is linear with regard to m, as the execution of any performed test is in particular linear with regard to m (the complexities with regard to m and n are independent). The pessimistic size of the conditional set in performed tests is equal to the maximal possible one: $n - 2$.

When we apply MMPC n times—one time for each variable, in order to detect the skeleton characterizing the faithful representation of the generative distribution, the performance of such procedure in all aspects remains the same as in the case of one MMPC run. So all the considered so far skeleton learning mechanisms are equally strong in the case of the scenario ST_1. Or it would be rather more adequate to say: equally weak. Unfortunately, the scenario ST_1 provides not enough simplifying assumptions in order to reduce the complexity of these methods.

The scenario SP_1, where we do not assume anything, is again indistinguishable from the scenario ST_1 in the context of the pessimistic performance of the learning methods. In this scenario also any strategy of the skeleton learning, included in SGS, PC, GS-GS or GS-IAMB, and the new one based on multiple runs of MMPC leads pessimistically to the exponential number of performed tests with respect to n, the exponential time complexity with regard to

n and linear with regard to m, and to the size of the maximal conditional set in performed tests equal to $n-2$.

5.2.11. Performance—Scenario ST_2

In the scenario ST_2 we assume that each node in the faithful representation of the generative distribution has at most k neighbors. Although the heuristic ordering of variables in MMPC is very strong, still there is no guarantee here that the $\mathbb{NB}$ set will not grow to an arbitrary large in the Grow phase. In particular, as we have mentioned it earlier, there is no guarantee to find for any processed variable X the conditional independence evidence of X and the target variable T corresponding to the conditional set included in the neighborhood of T in the faithful representation. As a consequence the performance of MMPC (as well as of multiple runs of MMPC in order to detect the skeleton characterizing the faithful representation) in this scenario remains unchanged with respect to the scenario ST_1.

So in the scenario ST_2 MMPC is very weak in fact, with the pessimistic performance the same as the most straightforward SGS algorithm. The PC for comparison requires performing $\mathcal{O}(n^{k+2})$ tests in order to retrieve the skeleton, calculating it in $\mathcal{O}(mn^{k+2})$ time, and with the maximal conditional set appearing in the tests equal to k. The GS-GS and GS-IAMB algorithms require performing only respectively $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ tests in order to learn the skeleton, and $\mathcal{O}(mn^3)$ time—although the pessimistic size of the conditional set appearing in tests is in both cases still equal to $n-2$.

5.2.12. Performance—Scenario SP_2

We will reach to the most interesting conclusions in the case of the scenario SP_2, where the only one assumption, the application of the Spirtes et al. (2000) convention, results in the abortion of all statistical tests for conditional independence corresponding to conditional sets of size exceeding $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$—with an automatic verdict of a conditional independence. Let us start with the analysis of the pessimistic number of performed tests:

- In the Grow phase of the MMPC algorithm the repetition of the loop can be automatically stopped in the moment when the size of the set $\mathbb{NB}$ exceeds the value b_m (in the next repetition the loop will be aborted according to the Spirtes et al. (2000) convention). So we perform such repetition at most $b_m + 1$ times, and in each repetition we perform for each of the $\mathcal{O}(n)$ candidate for inclusion to $\mathbb{NB}$ variables conditional independence tests with conditional sets corresponding to all subsets of $\mathbb{NB}$—in order to obtain the $MinAssoc_{\mathcal{P}}$ measures. The size of the set $\mathbb{NB}$ is bounded by $b_m + 1$, so for each of these $\mathcal{O}(n)$ variables we perform $\mathcal{O}(2^{b_m})$ tests, that is $\mathcal{O}(m)$ tests. There is one more test at the end of each repetition—deciding whether we stop the whole loop. However, it in fact does not have to be even calculated, as the decision regarding the considered here conditional independence is an immediate consequence of the obtained maximal $MinAssoc_{\mathcal{P}}$ value—whether the corresponding to it with respect to Equation 2.2.6.1 p-value is above or below the significance level.

  Summarizing, in the Grow phase we pessimistically perform $\mathcal{O}(mb_m n)$ tests, that is $\mathcal{O}((m \log_2 m)n)$ tests.

- In the Shrink phase we proceed one time each of the $\mathcal{O}(b_m)$ variables included in the set $\mathbb{NB}$ after the Grow phase, and for each such variable we perform $\mathcal{O}(2^{b_m})$ tests for conditional sets corresponding to all subsets of $\mathbb{NB}$—thus $\mathcal{O}(m)$ tests. So in the Shrink phase we pessimistically perform $\mathcal{O}(m \log_2 m)$ tests.

Overall in the MMPC method we perform pessimistically $\mathcal{O}((m \log_2 m)n)$ tests—so in order to calculate the whole skeleton we need to perform pessimistically $\mathcal{O}((m \log_2 m)n^2)$ tests.

The time complexity of each performed test is pessimistically $\mathcal{O}(mb_m)$, that is $\mathcal{O}(m \log_2 m)$—as each such test is performed with regard to the conditional set of limited by b_m size. As the result the time complexity of the MMPC method is $\mathcal{O}((m \log_2 m)^2 n)$—and so the time complexity of obtaining with the assist of MMPC the skeleton is $\mathcal{O}((m \log_2 m)^2 n^2)$.

As we have already stated it, the maximal size of the conditional set in performed tests for conditional independence is equal to b_m.

The most significant from practical point of view is exactly the considered here scenario SP_2, as it reflects well the situation which we face in the reality. What is important, the performance of MMPC is here comparable to the previous less aggressive, less ambitious, but also in general less time consuming approaches.

In the case of the pessimistic number of performed tests let us remind that in the SGS and PC algorithms this pessimistic number is polynomial with regard to both m and n, but the main problem is that the exponent in both cases depends directly on the second dimension of the input table. In order to obtain the skeleton with the assist of the GS-GS approach it is pessimistically required to perform $\mathcal{O}((m \log_2 m)n + n^2)$ tests, while in order to achieve this with the GS-IAMB method we need to perform $\mathcal{O}((m \log_2 m)n + (\log_2 m)n^2)$ tests. The pessimistic number of performed tests in the case of MMPC applied to learn the skeleton—$\mathcal{O}((m \log_2 m)n^2)$—is worse than in the case of both GS-GS and GS-IAMB methods, however at least with regard to separately considered dimensions m and n this pessimistic number is the same scalable in all these three approaches.

With respect to the time complexity the SGS and PC methods bind in similar relation dimensions m and n as in the case of the pessimistic number of performed tests. The time complexity of applying both the GS-GS and GS-IAMB method in order to calculate only the skeleton is $\mathcal{O}((m \log_2 m)n^2 + (m \log_2 m)^2 n)$. So again, the time complexity of obtaining the skeleton with the assist of MMPC—$\mathcal{O}((m \log_2 m)^2 n^2)$—is slightly worse than in the case of GS-GS and GS-IAMB—although with regard to separately considered both dimensions all these three algorithms have the same complexity.

With respect to the reliability all so far considered methods: SGS, PC, GS-GS, GS-IAMB and MMPC considered in the context of the skeleton learning are equally strong—for each of them the pessimistic size of the conditional set in performed tests is equal to b_m.

5.2.13. Performance—Summary

Summarizing the whole above analysis we should point it out that the side efficiency loss effect of the enormous effort in the MMPC method in order to as well as possible order the process of the variables inclusion in the Grow phase is not so painful. Although in the scenario ST_2 this side effect is mostly visible, in the most important from practical view scenario SP_2 the efficiency of MMPC is really good, almost as good as in the case of GS-GS and GS-IAMB. This efficiency is especially good if we keep in mind that MMPC is able to achieve, although not in the considered here worst case scenario, but in practice an incomparably better reliability of performed tests than its predecessors, and as the result much better accuracy of whole learning. We again refer an interested reader to the Tsamardinos et al. (2003b) work, where the empirical evaluation of the MMPC and MMMB mechanisms clearly shows the superiority of the Max-Min strategy over the previous approaches.

5.3. Max-Min Hill-Climbing

We have reached the point where the most sophisticated and successful among all considered in this dissertation state-of-the-art Bayesian network structure learning approaches is described. We present in this section the MMHC (Max-Min Hill-Climbing) algorithm, designed by Tsamardinos et al. (2006). This algorithm is a consequence of the intensive research in the field of local discovery methods, started few years earlier, with the Margaritis and Thrun (2000) work. It is in particular a beautiful complement and improvement of the previous notable solution in this area—the MMPC algorithm.

5.3.1. Symmetry Correction of MMPC

There are two main contributions of the MMHC algorithm. One resulting in the improvement of the preceding solution MMPC, and second resulting in its complement.

Let us start with the contribution related to the improvement of the MMPC algorithm. In the MMHC method in particular at its beginning the MMPC algorithm is invoked—one time for each attribute of the input dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$. As we have discussed it in the previous section, we can obtain in this way some approximation of the skeleton characterizing the class of faithful representations for the generative distribution of the sample $\mathcal{D}$—assuming it exists.

Some problem with the MMPC mechanism however was its luck of complete theoretical soundness. According to what we have stated previously subject to the appropriate theoretical conditions MMPC is guaranteed to return the superset of the neighborhood in the perfect map of the target variable. And we have seen an example which proves that this method can potentially include to the result more variables than necessary. As a consequence the skeleton obtained with an assist of MMPC invoked for each variable might in such theoretical conditions contain more edges than the real skeleton characterizing the class of faithful representations.

The solution to this problem, known as the "symmetry correction", has turned out to be surprisingly simple and elegant—and the MMHC algorithm was the first method which has utilized this idea. Namely, this correction can be seen in Algorithm 5.3.1.1—corresponding to the first part of the MMHC method—the skeleton learning part.

Algorithm 5.3.1.1 The MMHC Bayesian network skeleton learning algorithm.

1. Let us initialize the skeleton $\mathcal{G}$ as an empty undirected graph on the vertex set $\mathcal{V}$.
2. For each variable $X \in \mathcal{V}$ invoke the algorithm MMPC in order to find the approximation of its neighborhood in the desired skeleton—let us denote it as $\mathbb{NB}_X$.
3. For each $X \in \mathcal{V}$, and for each $Y \in \mathbb{NB}_X$, if $X \in \mathbb{NB}_Y$ then connect with an edge nodes X and Y in $\mathcal{G}$.
4. Return the skeleton $\mathcal{G}$ characterizing the Markov equivalence class.

In the case when we assume, as previously in the case of the plain MMPC solution, that there exists a generative distribution $\mathcal{P}$ for the input sample $\mathcal{D}$ admitting a faithful representation, that we obtain a perfect knowledge about existing in $\mathcal{P}$ conditional independencies via performed statistical tests, and that the same statistical test is used in order to resolve such conditional relations as well as calculate the $Assoc_{\mathcal{P}}$ values in the MMPC subroutine, this

approach, in contrary to the plain MMPC application, is guaranteed to return the skeleton characterizing the class of perfect maps of $\mathcal{P}$. The explanation is following:

- Any pair of different vertices not connected in the skeleton $\mathcal{G}$—the result of Algorithm 5.3.1.1, are not connected in the desired skeleton of perfect maps of $\mathcal{P}$. The reason is that according to the partial theoretical soundness of the MMPC algorithm we have that after performing Step 2 for each $X \in \mathcal{V}$ there holds $\mathbb{NB}_X \supseteq NB_X^{\mathcal{P}}$. According to the mechanism of Algorithm 5.3.1.1, for any pair of different vertices $X, Y \in \mathcal{V}$ there holds that they will be not connected in $\mathcal{G}$ after performing Step 3 if and only if either $X \notin \mathbb{NB}_Y$ or $Y \notin \mathbb{NB}_X$ holds—and this implies that for sure either $X \notin NB_Y^{\mathcal{P}}$ or $Y \notin NB_X^{\mathcal{P}}$ holds, which in turn means that both these inclusions does not hold.

- Any pair of different vertices connected in the resultant skeleton $\mathcal{G}$ are connected in the desired skeleton of perfect maps of $\mathcal{P}$. Let us consider such pair of vertices $X, Y \in \mathcal{V}$ connected in $\mathcal{G}$. Let us assume that X and Y are not connected in the skeleton of perfect maps of $\mathcal{P}$. In such case, as we know it from the previous section, there holds the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S})$ for the subset of variables $\mathbb{S}$ satisfying either $\mathbb{S} \subseteq NB_X^{\mathcal{P}}$ or $\mathbb{S} \subseteq NB_Y^{\mathcal{P}}$. If $\mathbb{S} \subseteq NB_X^{\mathcal{P}}$ then clearly after performing Step 2 we will have that $Y \notin \mathbb{NB}_X$—as in the Shrink phase of the MMPC subroutine invoked for the variable X in the moment of testing the variable Y for exclusion from the current set $\mathbb{NB}$ (assuming it was added to $\mathbb{NB}$ in the Grow phase) this conditional independence will be catched, and as the result Y will be removed. Similarly, if $\mathbb{S} \subseteq NB_Y^{\mathcal{P}}$, after performing Step 2 we will have that $X \notin \mathbb{NB}_Y$. So in both cases, according to the "symmetry correction" performed in Step 3 there will for sure not be an edge in $\mathcal{G}$ between X and Y—which leads us to the contradiction.

5.3.2. Dealing with Neglected Tests

In practice, when the typical solution to deal with unreliable tests for conditional independence, that is the Spirtes et al. (2000) convention, is applied, similarly as in the case of the plain MMPC application in MMHC we assume a conditional independence whenever a given test is neglected. The reason is inherited directly from our considerations about the algorithm MMPC itself—as here it is in particular also used. Namely, the automatic assumption of the conditional independence prevents us from the uncontrolled growth of the approximated set of neighbors in the Grow phase of the MMPC subroutine.

5.3.3. Symmetry Correction Performance

What is, besides the theoretical soundness of Algorithm 5.3.1.1, its nice property is that it achieves the same pessimistic bounds regarding its performance in all scenarios and aspects as the analyzed in the previous section plain application of MMPC in the skeleton learning task, without the "symmetry correction", that is without checking in Step 3 for each $X \in \mathcal{V}$ and for each $Y \in \mathbb{NB}_X$ whether $X \in \mathbb{NB}_Y$—instead for each $X \in \mathcal{V}$ and for each $Y \in \mathbb{NB}_X$ immediately inserting the edge connecting X and Y to $\mathcal{G}$.

It is obvious that there is no any difference here with respect to the number of performed statistical tests and the pessimistic size of the conditional set in these tests—as the "symmetry correction" does not lead to performing any additional tests. With respect to the time complexity the only difference is related to the aforementioned checking of the condition $X \in \mathbb{NB}_Y$ for each $X \in \mathcal{V}$ and for each $Y \in \mathbb{NB}_X$ in Step 3. Let us briefly discuss this cost:

- In the case of the scenarios ST_1, ST_2 and SP_1 we know from the previous section that the time cost of Algorithm 5.3.1.1 without the "symmetry correction" is exponential with regard to n (the number of attributes of the input dataset $\mathcal{D}$), and linear with regard to m (the number of objects in $\mathcal{D}$). The additional inclusion checking in Step 3 obviously does not change this overall cost.

- In the case of the scenario SP_2 we know that the time complexity of Algorithm 5.3.1.1 without the "symmetry correction" is $\mathcal{O}((m \log_2 m)^2 n^2)$. Clearly the additional cost related to performing the "symmetry correction" does not increase this overall cost, as this additional cost is clearly equal here to $\mathcal{O}(b_m^2 n)$ where $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$ for some defined by a user constant c, that is $\mathcal{O}((\log_2 m)^2 n)$. Namely, in Step 3 for each variable X belonging to the set $\mathcal{V}$ of size n we check for each $Y \in \mathbb{NB}_X$, that is—according to the previous section—for at most $b_m + 1$ variables, whether $X \in \mathbb{NB}_Y$—which can be done directly in $\mathcal{O}(b_m)$ time.

5.3.4. Wise Hybridization

The second contribution of the MMHC algorithm is its second part where the obtained in the first part with the assist of Algorithm 5.3.1.1 skeleton is converted into a DAG representing the resultant Bayesian network structure. The authors have decided to apply here the score-based learning approach, where let us remind that the main aim is to search for the DAG maximizing some scoring criterion with respect to the input dataset. As we have shown it in Chapter 3 such task is NP-hard in general, and in practice most commonly some heuristic approaches are applied, in order to find at least a relatively highly scored DAG.

Nevertheless, besides the most commonly heuristic nature of the score-based Bayesian network structure learning methods, their usefulness in practice can be high, especially in the aspects which are the biggest challenges in the field of constraint-based learning approaches. For example, from the point of view of the constraint-based strategies the problem of finding all directions, or more generally all v-structures determining a desired Markov equivalence class, is typically harder to resolve than the problem of finding the skeleton characterizing this class. This hardness occurs with respect to both time efficiency and accuracy of obtained results.

As we have discussed it in Section 4.2 several additional problems with the learned set of v-structures appear in practice, from which the most obvious problem is the inconsistency of the obtained set, for example in the sense that depending on the method, it can imply a cyclic graph, in particular a graph with edges in both directions, or more generally it does not determine a unique Markov equivalence class.

As we have also mentioned it in Subsection 4.2.2 several standard techniques repairing such resultant inconsistent characterizations of the Markov equivalence classes are typically applied, usually determining some DAG on the basis of the imperfect knowledge about these characterizations gathered during the actual constraint-based learning. The biggest problem with these repairing techniques is that they are in fact forgetting about the whole knowledge which we have on the entry, that is about the input dataset, and they rely on the purely graph algorithms and the obtained usually inconsistent network structure characterization only. For example they try to remove all cycles from this representation, and such procedure is optimized with regard to the time efficiency, but without any respect to the influence of such modifications on the quality of the resultant network. In other words these repairing solutions are as efficient as they can be, but not accurate.

A very interesting hybrid solution connecting the field of constraint-based and score-

based methods can arise exactly in the considered here problem of directing appropriately the learned skeleton. The idea is very simple. Instead of performing with an assist of some score-based approach the search for the highly-scored DAG in the space of all possible DAGs—we can significantly limit first the entry problem. For example, we can apply first some constraint-based method in order to learn only the skeleton of the network, and then instead of performing further problematic in the case of these kind of methods directing of this graph we can rely here on some score-based method, which tries to find a highly scored DAG among graphs characterized by the learned earlier skeleton, or characterized by any subgraph of this skeleton. In this way we simplify significantly the learning problem of both constraint-based and score-based components of such solution, adopting each of them to the task in which they are more accurate and efficient.

The first notable solution where the above idea of applying a score-based method over an appropriately limited search space is considered was the SC (Sparse Candidate) algorithm (Friedman et al., 1999). In the SC method however the weak point is the first, skeleton learning procedure. Namely, this skeleton is achieved by heuristically determining the set of possible parents of each vertex, where each such set has the size limited by some chosen by a user constant.

In the MMHC approach the same idea as the one coined by Friedman et al. (1999) has been applied, but here the skeleton learning part is based on the extremely reliable and theoretically sound MMPC algorithm with the "symmetry correction"—the specialized solution designed exactly with the aim of finding such skeleton.

5.3.5. Score-Based Part Description

The whole MMHC procedure is presented in Algorithm 5.3.5.1.

Algorithm 5.3.5.1 The MMHC Bayesian network structure learning algorithm.

1. Run Algorithm 5.3.1.1 in order to determine the skeleton $\mathcal{G}$ operating on the set of nodes $\mathcal{V}$—attributes of an input dataset $\mathcal{D}$.
2. Initialize the DAG $\mathcal{H}$ as an empty graph operating on the set of nodes $\mathcal{V}$.
3. Repeat:
   - Determine the DAG $\mathcal{H}'$ which is the neighbor of $\mathcal{H}$ maximizing value $Score(\mathcal{D}, \mathcal{H}')$. The neighbor of $\mathcal{H}$ means any DAG which can be obtained from $\mathcal{H}$ after applying one of the operations: removing, reversing or adding one edge—provided that no cycle appears in the case of reversion or addition, and that any added edge exists in undirected version in $\mathcal{G}$.
   - If $Score(\mathcal{D}, \mathcal{H}') > Score(\mathcal{D}, \mathcal{H})$ assign $\mathcal{H} := \mathcal{H}'$, otherwise terminate the loop.
4. Return the DAG $\mathcal{H}$ as a resultant Bayesian network structure.

The second part of the MMHC name corresponds exactly to the second, score-based part of the method, as this is in fact the classical metaheuristic HC (Hill-Climbing), used here in order to greedily search for a highly scored DAG (local maximum) with respect to a very limited space of DAGs characterized by the skeleton included in the one obtained in the first part of MMHC—that is in Algorithm 5.3.1.1. The search is organized according to the quite straightforwardly defined neighborhood relation in the explored space of DAGs. Namely, for

each currently considered DAG we search for its neighbors resulting from applying one of the operations: removal, insertion or reversion of one edge. The scoring criterion *Score* applied in the HC procedure can be here arbitrary chosen by the user.

Pessimistically the HC subroutine can traverse through exponentially many DAGs before finding a local maximum. So this part of the algorithm might be potentially some bottleneck. However, in practice the HC part is not so problematic. The most time effort is usually related to determining the skeleton $\mathcal{G}$ in Step 1.

5.3.6. MMHC Practical Significance

HC is only a heuristic, so as a consequence the whole MMHC approach is also a heuristic. But the fusion of the advanced MMPC skeleton learning phase with the heuristic HC procedure determining the final DAG corresponding to some local maximum with regard to the scoring criterion in the limited with the obtained earlier skeleton space turns out to be a good choice, especially in practice. An interested reader is encouraged to analyze the empirical evaluation reported by Tsamardinos et al. (2006). The MMHC approach is compared here with many other state-of-the-art Bayesian network structure learning approaches on the number of datasets derived from the well known benchmark Bayesian networks, in several aspects reflecting the efficiency and accuracy of the performed learning. MMHC wins in almost all these comparisons. In particular there was only one other algorithm which was able to achieve in some scenarios better accuracy of the learned model, however with the enormous time cost—the GES (Greedy Equivalent Search) exact score-based learning method (Chickering, 2002). We will talk about both these very famous methods a bit more in Chapter 7, where the wide empirical comparison of the proposed in this dissertation Bayesian network structure learning mechanism with the MMHC and GES methods is presented and discussed.

Chapter 6

Local Bayesian Networks Aggregation—Theoretical Analysis

This chapter includes the main results of this dissertation. After some introductory section presenting some motivations of the whole presented in the remaining part of this dissertation research, and placing it with respect to the related branch of local learning algorithms, a theoretical statement giving a simple recipe how to induce a global faithful representation from appropriate, corresponding to Markov blankets of all attributes, local faithful representations is introduced and proved. The practical realization of this induction methodology—the LBNA algorithm—is presented and analyzed at the last part of this chapter.

6.1. Motivation

Before going into any details we firstly coin in this section the basic idea standing behind the whole proposed solution. We also summarize the whole background of other related solutions, including also some not mentioned so far subbranches of local discovery learning methods. We finally place the proposed in this dissertation and in particular in this chapter approach in this whole field, and we indicate the novelty aspects.

6.1.1. LBNA Brief Characterization

As it has been mentioned in the introduction, this work presents some methodology of induction a global perfect map—based on all attributes, from local perfect maps—based on appropriate subsets of attributes. This statement is actually a simplification. What is in fact induced here is a Markov equivalence class of global perfect maps given the Markov equivalence class of local perfect maps for each considered subset of attributes. We assume here that both the global, based on all the attributes, distribution and the local, based on the considered subsets of attributes, distributions admit a faithful representation. These equivalence classes—local and resultant global, are assumed to be represented exactly in the form described previously—that is as a skeleton and a set of v-structures.

The whole methodology is called LBNA (Local Bayesian Networks Aggregation). Informally, each LBNA instance uses two arbitrary chosen tools: a Markov blanket approximation method and a Bayesian network structure learning algorithm. These tools are used in order to obtain local Bayesian network structures over Markov blanket-based subsets of attributes. The main contribution of LBNA itself is the aggregation of this local knowledge, in order to obtain a global—based on all attributes—Bayesian network, or expressing it more generally,

obtaining the characterization of the class of perfect maps operating on all attributes. As it will be explained in details in the next section, the fundamental property of this aggregation is its theoretical soundness—in the sense that the aggregation of optimal local structures results in obtaining a global optimal structure.

6.1.2. GS Decomposition and Merging Adjustments

As we have also mentioned about it in the introduction, the LBNA methodology is inspired by the Margaritis and Thrun (2000) work, that is by the GS algorithm, and can be placed similarly as all the discussed earlier approaches like GS, IAMB, MMPC or MMHC in the branch of local learning algorithms. LBNA establishes the fork in this branch in its root—the GS algorithm—as the proposed here solution, although similarly as the remaining mentioned here methods arose from the GS ideas, provides an enhancement in a completely different direction.

The GS Bayesian network structure learning algorithm is an approach in which we infer a global Bayesian network from the knowledge of learned at the beginning Markov blankets of each variable, using appropriately to these blankets applied statistical tests for conditional independence. The IAMB algorithm, as well as its descendants in the branch of local learning algorithms like MMPC and MMHC have raised on the idea of first of all enhancing the reliability of the Markov blanket learning subroutine, and secondly reducing the size of the locally learned structures and appropriately adopting to this reduction the whole network structure learning process. This path has turned out to be a very successful one.

However, this is not the only one path arising from the root GS method—although definitely the most recognizable one. Some other directions have raised too. For example, instead of modifying the GS Markov blanket learning subroutine one can think about enhancing the GS procedure of using the learned blankets in the process of inferring the network structure. An interesting modification of the GS procedure of converting Markov blankets into a Bayesian network—although of the same pessimistic time complexity, but empirically more efficient and reliable in specific situations—was proposed by Pellet and Elisseeff (2008).

6.1.3. Different Idea—Recursive Autonomy Identification

There have raised on the basis of the GS ideas also some algorithms which have completely changed the nature of the structure decomposition. In the main path of local learning algorithms containing such solutions like GS, IAMB, MMPC and MMHC, but also in other paths containing for example the mentioned Pellet and Elisseeff (2008) approach, the structure of the algorithm on its highest level is identical. For each attribute of the entry dataset we learn the corresponding subset of attributes expressing some local structure around it, and then we employ an appropriate strategy in order to construct more efficiently and reliably on the basis of these subsets the global, operating on all attributes, Bayesian network structure. To some extend also the proposed in this work LBNA solution has on its highest level analogical construction.

Changing this local learning highest level scheme into some completely different has turned out to be very fruitful. Such completely new framework has been established by Yehezkel and Lerner (2005, 2009), in the proposed RAI (Recursive Autonomy Identification) algorithm. As the name suggests, RAI is a recursive algorithm. The idea here is to on the top level of this recursion determine the initial approximation of the network structure with an assist of statistical tests for conditional independence with conditional sets of size 0 (that is simply with the tests for independence), and then decompose the problem of further more accurate

learning of this structure into investigation of its appropriately chosen substructures. On each next level of this recursion the tests of a higher order are applied, namely in each next level there are performed tests for conditional independence corresponding to conditional subsets of size increased by 1.

The RAI algorithm is a very sophisticated constraint-based Bayesian network structure learning algorithm—and definitely one of the most notable in this field. It is a theoretically sound method aimed at the extreme reduction of the number of performed statistical tests—especially the reduction of the number of tests of the highest order, that is these least reliable. In the experiments reported by Yehezkel and Lerner (2009), performed on many datasets derived from known benchmark Bayesian networks, RAI is an undisputed winner among the number of compared state-of-the-art algorithms, including MMHC, in such aspects like the number of performed statistical tests and the execution time. The most competitive is here the MMHC method, which in particular is able to win with RAI with respect to the quality of the learned structure in the case of some datasets. The superiority of the MMHC approach over the remaining algorithms in several aspects is here confirmed.

Regardless of the great ideas coined in this work, resulting in the advanced and strong in many aspects algorithm, the comparison presented in this work seems to be however a bit unfair, and it does not assure in fact what is the real relation between the MMHC and RAI performance. The problem is that in the case of the MMHC algorithm authors report the results corresponding to the original MMHC implementation, where as the test of conditional independence the G^2 test is used with the fixed significance level equal to 5%. In the case of RAI a different test is used, and moreover the results of RAI correspond here to the a-priori optimized choice of the significance level in performed inside RAI tests for conditional independence, maximizing the BDeu scoring criterion of the result. Consequently, the comparison where in the case of both methods the same strategy of determining conditional independencies is applied would be very welcome—but it is not given in this work.

We have not decided to test RAI in the experiments evaluating the proposed LBNA methodology reported in Chapter 7. The only one implementation of RAI which we were able to find is the original authors implementation, written in the MATLAB language. MATLAB is not an open-source tool, but the main problem is that in this implementation there is no possibility of applying the standard G^2 conditional independencies testing strategy, which would be a crucial condition to perform a reasonable comparison with another based on this convention approaches.

6.1.4. LBNA Place and Novelty

LBNA is an enhancement in the field of Bayesian network structure local learning algorithms in the completely different direction. It is strongly related to such solutions like these proposed by Margaritis and Thrun (2000) or Pellet and Elisseeff (2008) of inferring a global Bayesian network from the knowledge of Markov blankets of each variable, using appropriately to these blankets applied statistical tests for conditional independence. In particular in LBNA a global network is induced also from Markov blankets and some additionally acquired related to these blankets knowledge. But the main difference, because of which the LBNA can be considered as a significantly different and more general approach, lies in this acquired additional knowledge.

This knowledge is—instead of the results of appropriately applied statistical tests—in the form of local Bayesian networks, which—as we will discuss it further—are learned on the subsets of variables of the form $\{\text{variable}\} \cup \{\text{Markov blanket of this variable}\}$. The main advantage of this situation is that we can now choose whatever Bayesian network structure learning method we want in order to obtain this local knowledge. We can rely for example on

some constraint-based algorithm, resulting in the local knowledge of more or less but similar nature to the one acquired with GS—as in both approaches statistical tests are the main source of knowledge. But instead of this we can use any other Bayesian network structure learning algorithm we prefer. For example we can apply locally some very advanced approach if we want to obtain the local knowledge of a very high quality—potentially higher than in the case of GS, or in contrary a very simple but fast algorithm if we need to deal with the case of Markov blankets of the irregular and potentially very big size.

The important aspect of the Bayesian network induction methods from Markov blankets given by Margaritis and Thrun (2000) or Pellet and Elisseeff (2008) is their good time scalability with respect to the number of attributes in the case of learned Markov blankets of the limited by some common constant size. As we could note it in previous chapters the limited size of learned Markov blankets is not uncommon. In particular all considered previously classical Markov blanket learning approaches return blankets of limited size in the case of the scenario SP_2 (introduced in Subsection 4.1.3). This scenario is a very common situation in practical applications. As it will turn out further, the LBNA approach is also very well scalable in such scenario of limited blankets.

6.2. Theoretical Foundations

In this section we present and prove the main theoretical result on the basis of which the LBNA solution is based. We also denote some possible modifications of the result, which from the theoretical point of view are not important, but may cause significant differences in the practical realizations of the result. Finally we state and discuss some hypothesis about the expected possibility of strengthening the result.

6.2.1. Core Proposition

For the sake of simplicity let us introduce some new notations. For any Markov equivalence class $\mathcal{C}$, which, according to Theorem 2.4.3.1, we represent as a skeleton and a set of v-structures, by $V(\mathcal{C})$ and $E(\mathcal{C})$ we will respectively understand the set of vertices and the set of undirected edges in the skeleton characterizing the class $\mathcal{C}$.

Let $\mathcal{P}$ be the joint probability distribution of some set of random variables $\mathcal{V}$. For any nonempty subset $\mathbb{X} \subseteq \mathcal{V}$ by $\mathcal{P}\,|_{\mathbb{X}}$ we will understand the joint probability distribution of the set of variables $\mathbb{X}$ implied by the distribution $\mathcal{P}$. Assume that the distribution $\mathcal{P}\,|_{\mathbb{X}}$ admits a faithful representation for some $\emptyset \neq \mathbb{X} \subseteq \mathcal{V}$. In such a case we will denote the Markov equivalence class of all perfect maps of $\mathcal{P}\,|_{\mathbb{X}}$ as $MC^{\mathcal{P}}_{\mathbb{X}}$. Let us also remind that the Markov blanket of some variable $X \in \mathcal{V}$ with regard to the distribution $\mathcal{P}$ we denote by $MB^{\mathcal{P}}_X$, and when we assume that $\mathcal{P}$ admits a faithful representation $MB^{\mathcal{P}}_X$ is unique for all $X \in \mathcal{V}$, according to Theorem 4.3.1.1.

The whole LBNA methodology is derived from the following observation.

Proposition 6.2.1.1 *Let $\mathcal{P}$ be the joint probability distribution of some set of random variables $\mathcal{V}$. Assume that the distributions $\mathcal{P}$ and $\mathcal{P}\,|_{\{X\}\cup MB^{\mathcal{P}}_X}$ for every $X \in \mathcal{V}$ admit a faithful representation. Let us introduce the notation $\Lambda = \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$. Then:*

(a) The skeleton characterizing $MC^{\mathcal{P}}_{\mathcal{V}}$ can be induced from the skeletons characterizing the elements of Λ: for each two different vertices $X, Y \in \mathcal{V}$ it holds that $(X, Y) \notin E(MC^{\mathcal{P}}_{\mathcal{V}})$ $\iff$ at least one of the following conditions is true:

- $X \notin MB_Y^{\mathcal{P}}$ *(equivalently:* $Y \notin MB_X^{\mathcal{P}}$*)*
- *there is at least one element* $\lambda \in \Lambda$ *such that* $X \in V(\lambda)$, $Y \in V(\lambda)$ *and* $(X,Y) \notin E(\lambda)$.

(b) The set of v-structures characterizing $MC_{\mathcal{V}}^{\mathcal{P}}$ *can be induced from the sets of v-structures characterizing the elements of* Λ*: for each three different vertices* $X, Y, Z \in \mathcal{V}$ *such that* $(X,Y) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$, $(X,Z) \in E(MC_{\mathcal{V}}^{\mathcal{P}})$ *and* $(Y,Z) \in E(MC_{\mathcal{V}}^{\mathcal{P}})$ *it holds that* (X,Z,Y) *is a v-structure in* $MC_{\mathcal{V}}^{\mathcal{P}} \iff (X,Z,Y)$ *is a v-structure in at least one element* $\lambda \in \Lambda$.

Proof

(a) $\Longrightarrow$ Assume that the first condition does not hold, that is $X \in MB_Y^{\mathcal{P}}$. We will prove that the second condition must hold in such case.
The assumption that $X \in MB_Y^{\mathcal{P}}$ implies also that $Y \in MB_X^{\mathcal{P}}$. More generally, we have here that $X \in MB_Y^{\mathcal{P}} \iff Y \in MB_X^{\mathcal{P}}$, because both cases are according to Theorem 4.3.1.2 equivalent to the situation where either $(X,Y) \in E(MC_{\mathcal{V}}^{\mathcal{P}})$ or there exists such $Z \in \mathcal{V}$ that (X,Z,Y) is a v-structure in $MC_{\mathcal{V}}^{\mathcal{P}}$.
Let us consider an arbitrary chosen perfect map of the distribution $\mathcal{P}$, that is any DAG $\mathcal{G} \in MC_{\mathcal{V}}^{\mathcal{P}}$. We have that $(X,Y) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$, so in particular X and Y are not connected by an edge in $\mathcal{G}$. This means according to Remark 2.4.4.3 that X and Y are d-separated in $\mathcal{G}$ by $\pi_X^{\mathcal{G}}$ or by $\pi_Y^{\mathcal{G}}$. Let us create the aliases S and T for the variables X and Y: $\{S,T\} = \{X,Y\}$, such that S and T are d-separated in $\mathcal{G}$ by $\pi_S^{\mathcal{G}}$. So we have $Ind_{\mathcal{G}}(S,T \mid \pi_S^{\mathcal{G}})$, which, according to Corollary 2.3.4.1, implies that $Ind_{\mathcal{P}}(S,T \mid \pi_S^{\mathcal{G}})$. Note that $\{S\} \cup \{T\} \cup \pi_S^{\mathcal{G}} \subseteq \{S\} \cup MB_S^{\mathcal{P}}$ ($T \in MB_S^{\mathcal{P}}$, because we have $X \in MB_Y^{\mathcal{P}}$ and $Y \in MB_X^{\mathcal{P}}$; $\pi_S^{\mathcal{G}} \subseteq MB_S^{\mathcal{P}}$ thanks to Theorem 4.3.1.1). This means that we have $Ind_{\mathcal{P}|_{\{S\} \cup MB_S^{\mathcal{P}}}}(S,T \mid \pi_S^{\mathcal{G}})$, which implies that $Ind_{\mathcal{H}}(S,T \mid \pi_S^{\mathcal{G}})$, where $\mathcal{H}$ is an arbitrary chosen perfect map of the distribution $\mathcal{P}\,|_{\{S\} \cup MB_S^{\mathcal{P}}}$ ($\mathcal{H} \in MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}}$). This in turn means according to Remark 2.4.4.1 that S and T are not connected by an edge in $\mathcal{H}$—so also $(S,T) \notin E(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$. As the result we can put $\lambda = MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}}$.

$\Longleftarrow$ First let us consider the case that $X \notin MB_Y^{\mathcal{P}}$. In such situation the immediate consequence of Theorem 4.3.1.2 is that $(X,Y) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$.
Now let us assume the second possibility: there is such $\lambda \in \Lambda$ that $X \in V(\lambda)$, $Y \in V(\lambda)$ and $(X,Y) \notin E(\lambda)$. This means that for any DAG $\mathcal{G} \in \lambda$ the following d-separation occurs for some $\mathbb{Z} \subset V(\lambda)$: $Ind_{\mathcal{G}}(X,Y \mid \mathbb{Z})$. The consequence is $Ind_{\mathcal{P}|_{V(\lambda)}}(X,Y \mid \mathbb{Z})$, and so $Ind_{\mathcal{P}}(X,Y \mid \mathbb{Z})$. As the result for any DAG $\mathcal{H} \in MC_{\mathcal{V}}^{\mathcal{P}}$ it holds that $Ind_{\mathcal{H}}(X,Y \mid \mathbb{Z})$, so according to Remark 2.4.4.1 X and Y are not connected by an edge in $\mathcal{H}$. Again the conclusion is that $(X,Y) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$.

(b) $\Longrightarrow$ In particular we have that $(X,Y) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$ and $X \in MB_Y^{\mathcal{P}}$—because we assume that (X,Z,Y) is a v-structure in $MC_{\mathcal{V}}^{\mathcal{P}}$—so from Part (a) (direction "$\Longrightarrow$") of the proof we know that there exists such $S \in \{X,Y\}$ that $X \in V(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$, $Y \in V(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$ and $(X,Y) \notin E(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$. We have that $(X,Z) \in E(MC_{\mathcal{V}}^{\mathcal{P}})$ and $(Y,Z) \in E(MC_{\mathcal{V}}^{\mathcal{P}})$—this means thanks to Theorem 4.3.1.2 that independently on the choice of $S \in \{X,Y\}$ it holds that $Z \in MB_S^{\mathcal{P}}$, so also $Z \in V(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$. Let us also notice that it must be $(X,Z) \in E(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$ and $(Y,Z) \in E(MC_{\{S\} \cup MB_S^{\mathcal{P}}}^{\mathcal{P}})$—otherwise we would deduce directly from Part (a) of this proposition that $(X,Z) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$ or $(Y,Z) \notin E(MC_{\mathcal{V}}^{\mathcal{P}})$.

What we are trying to reach is that (X, Z, Y) is a v-structure in $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$—the only remaining thing is to have appropriate directions of the edges. Let us prove it by contradiction.
Assume that (X, Z, Y) is not a v-structure in $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$. Let $\mathcal{G} \in MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$ be an arbitrary chosen perfect map of the distribution $\mathcal{P}\,|_{\{S\}\cup MB^{\mathcal{P}}_S}$. We have that X and Y are not connected by an edge in $\mathcal{G}$, while there is an edge between X and Z and between Y and Z in $\mathcal{G}$. Moreover (X, Z, Y) is not a v-structure in $\mathcal{G}$. So from Remark 2.4.4.2 we in particular obtain that for some $\mathbb{Z} \subset \{S\} \cup MB^{\mathcal{P}}_S$ satisfying $Z \in \mathbb{Z}$ it holds that $Ind_{\mathcal{G}}(X, Y \mid \mathbb{Z})$. This in turn implies $Ind_{\mathcal{H}}(X, Y \mid \mathbb{Z})$ for an arbitrary chosen DAG $\mathcal{H} \in MC^{\mathcal{P}}_{\mathcal{V}}$. This implication was already considered and explained in Part (a) "$\Longleftarrow$" of the proof. But from the other side we have that (X, Z, Y) is a v-structure in $\mathcal{H}$, which according to Remark 2.4.4.2 means that X and Y are not d-separated in $\mathcal{H}$ by any subset of vertices containing vertex Z. This is a contradiction with previously shown $Ind_{\mathcal{H}}(X, Y \mid \mathbb{Z})$.

$\Longleftarrow$ Consider arbitrary chosen DAGs $\mathcal{G} \in \lambda$ and $\mathcal{H} \in MC^{\mathcal{P}}_{\mathcal{V}}$. We have that (X, Z, Y) is a v-structure in $\mathcal{G}$, so from Remark 2.4.4.2 we know that for some $\mathbb{Z} \subset V(\lambda)$ satisfying $Z \notin \mathbb{Z}$ it holds that $Ind_{\mathcal{G}}(X, Y \mid \mathbb{Z})$. This implies $Ind_{\mathcal{H}}(X, Y \mid \mathbb{Z})$, which means (again Remark 2.4.4.2) that (X, Z, Y) is a v-structure in $\mathcal{H}$, and therefore also in $MC^{\mathcal{P}}_{\mathcal{V}}$.

■

It must be pointed out that the order of the $MC^{\mathcal{P}}_{\mathcal{V}}$ characterizing elements induction is important. Firstly we obtain the skeleton of $MC^{\mathcal{P}}_{\mathcal{V}}$, and thanks to this we can check only these triples of vertices for being a v-structure, for which the two undirected edges corresponding to the candidate v-structure can be found in the skeleton.

6.2.2. Modifications

The another important thing is that after carefully reading the proof of the above proposition it can be noticed that some slight modifications can be instantly proposed. Namely, both Parts (a) and (b) of the proposition would remain true if the phrase "*at least one element* $\lambda \in \Lambda$" we replaced with the phrase "*at least one element* $\lambda \in \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}, MC^{\mathcal{P}}_{\{Y\}\cup MB^{\mathcal{P}}_Y}\}$". This actually does not need any additional explanation. The proof of Proposition 6.2.1.1 simply includes the proof of such modified versions.

In theoretical conditions, that is assuming that a global perfect map together with all considered in Proposition 6.2.1.1 local perfect maps exists, and that we have a perfect knowledge about these local structures, all four possible versions of the proposition—the results of applying the above mentioned modifications to Part (a) or (b)—are equivalent in the sense that they produce exactly the same resultant global model. However, in practice, where the assumptions are in fact unverifiable and learned local structures can be seen just as some approximations, of course differences in a resultant network can occur among these versions.

This work is focused on Proposition 6.2.1.1 in its original form, that is without any modifications. The reason why we choose such version lies mainly in the procedure of skeleton learning. Let us notice that the original recipe given in Part (a) of Proposition 6.2.1.1 will always produce a skeleton at least as sparse as the recipe with the mentioned modifications (the first skeleton must be simply a subgraph of the second one). We prefer a sparser result, that is why we stick to the original procedure. Of course there is no any reason to claim that the modification is useless. We plan to investigate these and possibly also another modifica-

tions (for example some hybrid approaches) of the merging procedure in the future. Here we are basing only on the pure Proposition 6.2.1.1.

6.2.3. Hypothetical Strengthening

Let us add one more comment regarding the introduced here proposition. We believe that this result can be strengthen. Namely, we believe that the whole proposition remains true if we only assume that the whole joint probability distribution $\mathcal{P}$ admits a faithful representation, and when we do not assume anything about the nature of the local distributions $\mathcal{P}\mid_{\{X\}\cup MB_X^{\mathcal{P}}}$ for all $X \in \mathcal{V}$. The reason is that we believe that the faithfulness of the local distributions—results of truncating the global distribution to appropriate subsets of variables corresponding to Markov blankets—follows from the faithfulness of the global distribution. Namely, we believe that the following hypothesis holds.

Hypothesis 6.2.3.1 *Assume that the joint probability distribution $\mathcal{P}$ of some set of random variables $\mathcal{V}$ admits a faithful representation. Let $X \in \mathcal{V}$. Then the distribution $\mathcal{P}\mid_{\{X\}\cup MB_X^{\mathcal{P}}}$ admits a faithful representation.*

What is for sure known is that the more general statement where we would replace the subset of the form $\{X\} \cup MB_X^{\mathcal{P}}$ with any other subset is not always true. We will explain it in the moment. But first we need some more observations.

6.2.4. Local Structures Properties

Below we present and prove a simple observation which as we will further see enables to deduce the faithful representation over the subset of variables assuming it exists from the knowledge of the faithful representation of the whole set of considered variables.

Remark 6.2.4.1 *Let $\mathcal{P}$ be the joint probability distribution of some set of random variables $\mathcal{V}$, admitting a faithful representation, where $\mathcal{G}$ is some perfect map of $\mathcal{P}$. Let $\mathbb{S}$ be an arbitrary nonempty subset of $\mathcal{V}$, and $\mathcal{H}$ be an arbitrary DAG operating on the set of nodes $\mathbb{S}$. Then the following two conditions are equivalent:*

- *The distribution $\mathcal{P}\mid_{\mathbb{S}}$ admits a faithful representation, where $\mathcal{H}$ is its exemplary perfect map.*
- *For any different $X, Y \in \mathbb{S}$ and $\mathbb{Z} \subseteq \mathbb{S} \setminus \{X, Y\}$ there holds $Ind_{\mathcal{G}}(X, Y \mid \mathbb{Z})$ if and only if there holds $Ind_{\mathcal{H}}(X, Y \mid \mathbb{Z})$.*

Proof The following sequence of equivalent statements leads from the first point to the second:

1. The distribution $\mathcal{P}\mid_{\mathbb{S}}$ admits a faithful representation, where $\mathcal{H}$ is its exemplary perfect map.
2. For any different and mutually exclusive subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subseteq \mathbb{S}$ where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty there holds $Ind_{\mathcal{H}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ if and only if there holds $Ind_{\mathcal{P}\mid_{\mathbb{S}}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$.
3. For any different and mutually exclusive subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subseteq \mathbb{S}$ where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty there holds $Ind_{\mathcal{H}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ if and only if there holds $Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$.
4. For any different and mutually exclusive subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subseteq \mathbb{S}$ where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty there holds $Ind_{\mathcal{H}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ if and only if there holds $Ind_{\mathcal{G}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$.

5. For any different $X, Y \in \mathbb{S}$ and $\mathbb{Z} \subseteq \mathbb{S} \setminus \{X, Y\}$ there holds $Ind_{\mathcal{G}}(X, Y \mid \mathbb{Z})$ if and only if there holds $Ind_{\mathcal{H}}(X, Y \mid \mathbb{Z})$.

The equivalence of 1 and 2 results from the definition of the perfect map/faithful representation (Definition 2.4.2.1). The equivalence of 2 and 3 is obvious. The equivalence of 3 and 4 is a consequence of the fact that $\mathcal{G}$ is a perfect map of $\mathcal{P}$.

Finally, the equivalence of 4 and 5 can be seen as follows. The implication $4 \implies 5$ is obvious. The implication $5 \implies 4$ is the consequence of the d-separation nature—that is Definition 2.3.4.3. Namely it can be instantly noticed from this definition that for any different and mutually exclusive subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subseteq \mathbb{S}$ where $\mathbb{X}$ and $\mathbb{Y}$ are nonempty there holds $Ind_{\mathcal{G}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$ if an only if there holds $Ind_{\mathcal{G}}(X, Y \mid \mathbb{Z})$ for all $X \in \mathbb{X}$ and $Y \in \mathbb{Y}$—and the analogical statement is also true for the graph $\mathcal{H}$. ■

The following remark shows that for any DAG there exists a distribution perfectly consistent with it. As a consequence In Remark 6.2.4.1 we can forget about the distribution $\mathcal{P}$—we can think about the graph $\mathcal{G}$ only, as the corresponding distribution for sure exists.

Remark 6.2.4.2 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary DAG. Then it is possible to define the set of random variables $\mathcal{V}$ corresponding to the nodes of the graph $\mathcal{G}$ such that the joint probability distribution of $\mathcal{V}$ admits a faithful representation, where an exemplary perfect map is $\mathcal{G}$.*

The proof of the above remark is an immediate consequence of the theoretical results provided by Meek (1995)—to which we will return in Section 7.1.4.

6.2.5. Hypothesis—Markov Blanket Restriction Importance

In order to show the example where the distribution of the subset of variables does not admit a faithful representation, although the distribution of all variables admits it, let us consider the following example.

Example 6.2.5.1 *Let us assume that the joint probability distribution $\mathcal{P}$ of some set of random variables $\{A, B, C, D, E\}$ admits a faithful representation, where an exemplary perfect map $\mathcal{G}$ for $\mathcal{P}$ is given in Figure 6.2.5.1 (according to Remark 6.2.4.2 such distribution exists).*

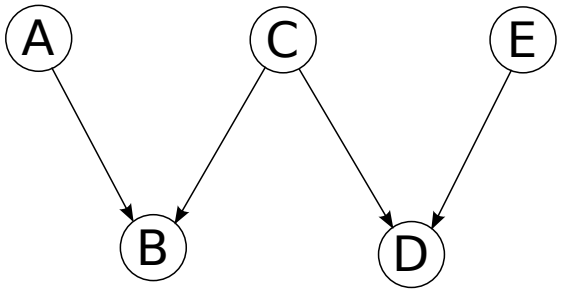


Figure 6.2.5.1: An exemplary perfect map $\mathcal{G}$ of some joint probability distribution $\mathcal{P}$ operating on the set of variables $\{A, B, C, D, E\}$.

Now, let us consider the distribution $\mathcal{P}\,|_{\{A,B,D,E\}}$. Assume that this distribution admits a faithful representation. In such case there exists some perfect map $\mathcal{H}$ of it, that is a DAG which

contains d-separations exactly corresponding to the conditional independencies occurring in this distribution. According to Remark 6.2.4.1 $\mathcal{H}$ contains exactly the same d-separations as $\mathcal{G}$ among the set of variables $\{A, B, D, E\}$. So in particular any edge occurring in the skeleton of $\mathcal{G}$ between any two of the variables from the set $\{A, B, D, E\}$ must also occur in the skeleton of $\mathcal{H}$. Namely, if we consider any two different $X, Y \in \{A, B, D, E\}$ such that X and Y are connected by an edge in $\mathcal{G}$, then according to Remark 2.4.4.1 X and Y are not d-separated in $\mathcal{G}$ by any subset of remaining variables, which means that also in $\mathcal{H}$ these two nodes are not d-separated by any subset of variables—and here we again adopt Remark 2.4.4.1 in order to conclude that X and Y are connected in $\mathcal{H}$.

So what we for sure know now is that the pairs (A, B) and (D, E) are connected by an edge in $\mathcal{H}$. Let us now investigate all the remaining connections in $\mathcal{H}$:

- *Notice that each of the pairs (A, D), (A, E) and (B, E) is d-separated by the empty set in $\mathcal{G}$, and all these d-separations occur also in $\mathcal{H}$—which means that each of these pairs is not connected in $\mathcal{H}$.*
- *The last not considered pair (B, D) is clearly not d-separated in $\mathcal{G}$ by any subset of two remaining occurring in $\mathcal{H}$ nodes: A and E. So according to Remark 6.2.4.1 such d-separation does not occur in $\mathcal{H}$—and as a consequence according to Remark 2.4.4.1 B and D are connected by an edge in $\mathcal{H}$.*

We have already induced the whole skeleton of the graph $\mathcal{H}$. It is illustrated in Figure 6.2.5.2.

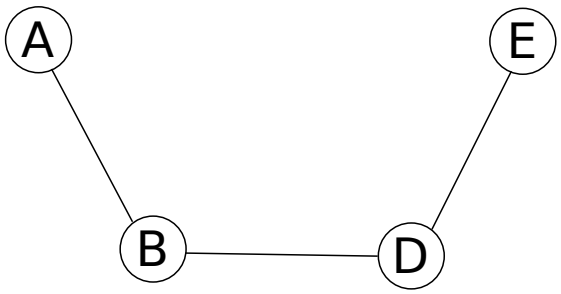


Figure 6.2.5.2: The skeleton of a hypothetic perfect map $\mathcal{H}$ of the distribution $\mathcal{P}|_{\{A,B,D,E\}}$.

In the similar fashion as the connections in $\mathcal{H}$ we should be able to deduce all v-structures appearing in it. In particular let us notice that:

- *We have $Ind_{\mathcal{H}}(A, D, \emptyset)$, so according to Remark 2.4.4.2 the triple (A, B, D) is a v-structure in $\mathcal{H}$.*
- *We have also $Ind_{\mathcal{H}}(B, E, \emptyset)$, so also the triple (B, D, E) is a v-structure in $\mathcal{H}$.*

But obviously both triples (A, B, D) and (B, D, E) cannot be v-structures in $\mathcal{H}$ as this would lead to the bidirectional edge (B, D). We have reached to the contradiction—which means that the distribution $\mathcal{P}|_{\{A,B,D,E\}}$ does not admit a faithful representation.

6.2.6. Hypothesis—Premise

A reader might be interested why do we believe in Hypothesis 6.2.3.1. The main reason is that we did not find so far any example which would reject this hypothesis. The methodology

of searching for such rejection is in our case following: define any DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which corresponds to the perfect map of some distribution of its nodes, consider for any of its nodes X a subset of nodes $\mathbb{S}$ containing X and its Markov blanket (the set of parents, children and parents of children in $\mathcal{G}$—according to the characterization given in Theorem 4.3.1.1), and reconstruct the hypothetical local faithful representation of the distribution truncated to $\mathbb{S}$—that is the characterizing it skeleton and set of v-structures —with an assist of Remark 6.2.4.1 and the most straightforward SGS/PC discovery strategy basing on Remarks 2.4.4.1 and 2.4.4.2, as in the example above. If the obtained local representation is consistent, that is it indeed represents some Markov equivalence class, and moreover it contains exactly the same d-separations as the original whole graph within the subset of the local structure attributes—then the contradiction is not found.

We did not perform any regular experiments here, but for the number of randomly picked local Markov blanket-based structures corresponding to several benchmark Bayesian networks defined in Section 7.1.3 we were not able to find any rejection of Hypothesis 6.2.3.1. We are planning to investigate this problem further in the future, especially with the aim of proving this hypothesis.

Some example confirming this hypothesis will be in particular given at the end of the following section.

6.3. LBNA—Theoretical Version

In this section we present the theoretical version of the LBNA (Local Bayesian Networks Aggregation) algorithm. We explain the equivalence of the procedure to the recipe given in Proposition 6.2.1.1. We investigate the efficiency of the proposed mechanism. Finally we visualize the whole procedure on the example run.

6.3.1. Description

LBNA is a relatively straightforward realization of Proposition 6.2.1.1. Namely, the algorithm employs the whole recipe included in this proposition in order to—subject to the assumptions provided in Proposition 6.2.1.1—infer the global faithful representation from the local ones.

In the presented here procedure we in particular assume that we have on the entry a perfect knowledge about the Markov equivalence classes of faithful representations $\Lambda = \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$ for the considered joint probability distribution $\mathcal{P}$ of some set of random variables $\mathcal{V}$. Thus according to Proposition 6.2.1.1 we know exactly how to induce from them the global Markov equivalence class of faithful representations $MC^{\mathcal{P}}_{\mathcal{V}}$. Algorithm 6.3.1.1 is one of the possible realizations of this induction. As we will explain it in the moment it is an equivalent procedure to the one included in the proposition.

This procedure corresponds to many earlier presented in this dissertation methods, where the proposed solution is applicable only in theory, while in practice some additional repairing procedures are required in order to assure the consistency of the returned result. As usually in such cases the inconsistency may arise in the case of learned v-structures. In the next section we propose and analyze some repairing procedure resolving this practical issue.

As one can notice the main difference between the recipe included in Proposition 6.2.1.1 and the procedure given in Algorithm 6.3.1.1 is that in the latter case we build both the global skeleton and set of v-structures by investigating consecutively each next local structure. Namely we proceed all these local structures three times—two times in order to retrieve the global skeleton and one time in order to retrieve the set of v-structures. Such limitation to

iterations among these local structures and deriving each required global information relaying on its local properties might be important from the point of view of the efficiency, which we will see soon.

Algorithm 6.3.1.1 The LBNA algorithm—the theoretical version.

1. Initialize the skeleton $\mathcal{G}$ characterizing $MC^{\mathcal{P}}_{\mathcal{V}}$ as an empty graph.
2. For each $X \in \mathcal{V}$ add to the skeleton $\mathcal{G}$ (in undirected versions) all and only these edges appearing in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$, which are incident there to the vertex X.
3. (The optional step) For each $X \in \mathcal{V}$ remove from $\mathcal{G}$ any edge between two vertices appearing in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ and not connected by an edge there.
4. Initialize the set of v-structures $\mathcal{C}$ characterizing $MC^{\mathcal{P}}_{\mathcal{V}}$ as an empty set.
5. For each $T \in \mathcal{V}$ and for each v-structure (X, Z, Y) included in $MC^{\mathcal{P}}_{\{T\}\cup MB^{\mathcal{P}}_T}$, if the pair (X, Z) and (Y, Z) is connected in $\mathcal{G}$ while the pair (X, Y) is not connected in $\mathcal{G}$ add the triple (X, Z, Y) to the set $\mathcal{C}$.
6. Return the skeleton $\mathcal{G}$ and the set of v-structures $\mathcal{C}$ as the characterization of $MC^{\mathcal{P}}_{\mathcal{V}}$.

6.3.2. Correctness Analysis

However, it is not hard to see that in fact Algorithm 6.3.1.1 is equivalent to the recipe included in Proposition 6.2.1.1, that is it leads to the same result subject to the assumption that we have the perfect knowledge about local faithful representations:

- After performing Steps 1 and 2 no more edges will be required in the global model, as it is easy to see from Part (a) of Proposition 6.2.1.1 that each inserted to the skeleton characterizing $MC^{\mathcal{P}}_{\mathcal{V}}$ edge must be incident to some vertex $S \in \mathcal{V}$ in $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$. Namely, for each $S \in \mathcal{V}$ we have that each $X \notin MB^{\mathcal{P}}_S$ is not connected by an edge with S in $MC^{\mathcal{P}}_{\mathcal{V}}$, and moreover also any $X \in MB^{\mathcal{P}}_S$ such that X and S are not connected by an edge in $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$ is not connected by an edge with S in $MC^{\mathcal{P}}_{\mathcal{V}}$.

  Moreover, it can be easily noticed that after performing Step 2 the skeleton $\mathcal{G}$ is exactly the desired skeleton of the global faithful representation. The reason follows from Theorem 4.3.4.1. According to this theorem any two different vertices X and Y not connected in the global perfect map are d-separated by both some subset of variables included in $MB^{\mathcal{P}}_X$ and some subset of variables included in $MB^{\mathcal{P}}_Y$. So in particular there holds both the conditional independence $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S}_1)$ for some subset $\mathbb{S}_1 \subseteq MB^{\mathcal{P}}_X$ and $Ind_{\mathcal{P}}(X, Y \mid \mathbb{S}_2)$ for some subset $\mathbb{S}_2 \subseteq MB^{\mathcal{P}}_Y$. If $Y \in MB^{\mathcal{P}}_X$ (equivalently $X \in MB^{\mathcal{P}}_Y$) then both these independencies occurs obviously in the respectively truncated distributions $\mathcal{P}\,|_{\{X\}\cup MB^{\mathcal{P}}_X}$ and $\mathcal{P}\,|_{\{Y\}\cup MB^{\mathcal{P}}_Y}$—so in particular the corresponding d-separations occurs respectively in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ and $MC^{\mathcal{P}}_{\{Y\}\cup MB^{\mathcal{P}}_Y}$—which according to Remark 2.4.4.1 means that X and Y are not connected by edge in the skeletons characterizing both these local classes.

  The conclusion is that after performing Step 2 for any pair of vertices (X, Y) such that $(X, Y) \notin E(MC^{\mathcal{P}}_{\mathcal{V}})$ it holds that:

– Either X does not belong to $V(MC^{\mathcal{P}}_{\{Y\}\cup MB^{\mathcal{P}}_Y})$ (equivalently Y does not belong to $V(MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X})$)—and in such case the pair (X,Y) will be for sure not included to the global structure.
– Or Y is not a neighbor of X in the skeleton characterizing $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ and X is not a neighbor of Y in the skeleton characterizing $MC^{\mathcal{P}}_{\{Y\}\cup MB^{\mathcal{P}}_Y}$—so clearly the edge (X,Y) will not be included to the global model also in such case.

- Step 3 is an optional step. It does not change absolutely anything in the considered here theoretical conditions. Simply no modifications of the skeleton appear after performing it. It could not exclude any edge which should not be excluded—as all potentially excluded here edges would satisfy the second condition for being excluded described in Part (a) of Proposition 6.2.1.1. And as we know in Step 2 all unnecessary edges have been already excluded—so Step 3 does not modify the skeleton.

- Step 5 is equivalent to the recipe included in Part (b) of Proposition 6.2.1.1—which is an obvious observation.

A reader might be potentially interested why Step 3 is included in Algorithm 6.3.1.1 if it is completely unnecessary—as the whole method remains theoretically sound without performing it. The reason is that although the optional Step 3 does not change anything in the theoretical conditions, it can potentially results in significant differences in the practical application of this algorithm, where on the entry some learned local structures, typically incorrect, are only available. In such case the strategy in Step 3 of the additional search for exclusion of some edges might potentially lead to the significantly sparser learned global structure. The sparsity of the Bayesian network is generally a desired property, so we in practice stick to such extended version of the LBNA algorithm. Note that this is a consistent choice with our previous considerations about the practical interpretation of Proposition 6.2.1.1 included in Subsection 6.2.2.

6.3.3. Performance Analysis

Let us now analyze Algorithm 6.3.1.1 from the point of view of its efficiency. In the case of previous algorithms we have analyzed the performance of each such method with regard to the performed statistical tests for conditional independence also. Here we do not perform any such tests, so the only reasonable aspect to analyze is the time complexity of the approach. Moreover, the scenarios ST_1, ST_2, SP_1 and SP_2 considered in the previous chapters were designed for the case of constraint-based algorithms, where in particular some assumptions regarding the performed tests for conditional independence and the input dataset are included. Here we do not perform any tests, and moreover, we do not have as an input any dataset. The only input for the LBNA algorithm are the local structures $\Lambda = \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$. So the completely different scenarios should be for such case designed. A similar to the previous analysis of the whole practical version of LBNA, which is more similar with respect to the inputs to the the previous solutions, will be performed in the next section. Here, let us consider only the following two scenarios:

- The first scenario corresponds to no assumptions—where in particular the local structures can be arbitrary large. In such case the time complexity of Algorithm 6.3.1.1 with regard to $n = |\mathcal{V}|$ can be summarized as follows:

- In Step 2 for each of n variables X we pessimistically process $\mathcal{O}(n)$ neighbors of X in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$. So this step costs us pessimistically $\mathcal{O}(n^2)$ time.
- Step 3 costs us pessimistically $\mathcal{O}(n^3)$ time, as for each of the n variables X we pessimistically process $\mathcal{O}(n^2)$ pairs of vertices included in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ and not connected there.
- In Step 5 we are searching for each of the n variables X for all v-structures appearing in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$, that is for all the suitable triples of nodes included there—which costs us $\mathcal{O}(n^3)$ time. So the time complexity of this step is $\mathcal{O}(n^4)$.

Summarizing, the time complexity of LBNA in the considered here scenario of no assumptions is $\mathcal{O}(n^4)$.

- The second scenario reflects the situation where each of the input local structures has bounded by some common constant k size. Namely, we assume that for each $X \in \mathcal{V}$ there holds $|V(MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X})| \leqslant k$. Let us analyze in this case the time complexity of LBNA both with regard to n and k—as such detailed analysis will be useful in the next section. Basing on the previous point it is easy to conclude that:
  - Step 2 costs $\mathcal{O}(nk)$ time.
  - Step 3 costs $\mathcal{O}(nk^2)$ time.
  - Step 5 costs $\mathcal{O}(nk^3)$ time.

  So the overall time complexity of LBNA in the second scenario of limited local structures is $\mathcal{O}(nk^3)$.

As we can see, in the case of local structures of the size bounded by some common constant the LBNA theoretical version has linear with regard to n time complexity—which is its important property. This in particular means that whenever such case holds the time cost of the LBNA merging procedure of local structures can be negligible comparing to all executed in practice preceding to LBNA local structures learning procedures—especially learning the Markov blankets of each variable, where any typical solution for this problem has at least quadratic with regard to n time complexity.

6.3.4. LBNA Example—Prerequisites

We finish this section with an exemplary visualization of Algorithm 6.3.1.1.

Example 6.3.4.1 *Let us consider the joint probability distribution $\mathcal{P}$ operating on the set of random variables $\mathcal{V} = \{A, B, D, E, L, S, T, X\}$, admitting a faithful representation, where the DAG $\mathcal{G}$ presented in Figure 6.3.4.1 is its exemplary perfect map (according to Remark 6.2.4.2 such distribution exists). This Bayesian network structure is not accidental—it is the structure of the very commonly exposed in the literature Bayesian network, called Asia (Lauritzen and Spiegelhalter, 1988). It is in fact an extended version of the network presented in Figure 2.3.3.1 representing the lung x-ray diagnoses of patients, here including also additional factors, in particular whether the patient has recently visited Asia. We do not go further into any details of this network here, as it will be not important for us.*

We can determine all the local faithful representations $\Lambda = \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$. According to Hypothesis 6.2.3.1 these representations exists.

The methodology of determining the local structure $MC^{\mathcal{P}}_{\{T\}\cup MB^{\mathcal{P}}_T}$ for each $T \in \mathcal{V}$ is following:

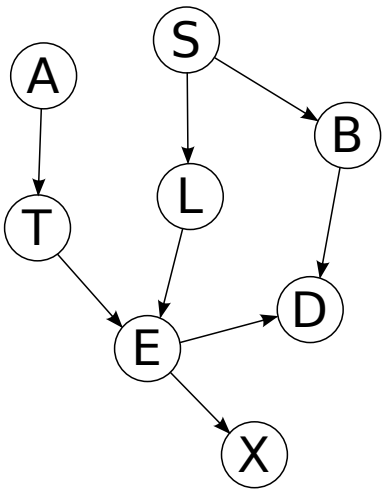


Figure 6.3.4.1: An exemplary perfect map $\mathcal{G}$ of the considered in Example 6.3.4.1 distribution $\mathcal{P}$.

- *Determine the Markov blanket of* $MB_T^{\mathcal{P}}$ *as the set of parents, children and parents of children of* T *in* $\mathcal{G}$.
- *Initialize the skeleton* $\mathcal{H}$ *characterizing* $MC^{\mathcal{P}}_{\{T\}\cup MB_T^{\mathcal{P}}}$ *as a clique.*
- *For each pair of vertices* $X, Y \in \{T\} \cup MB_T^{\mathcal{P}}$ *remove the edge* (X, Y) *from* $\mathcal{H}$ *if there exists a subset* $\mathbb{S}$ *of remaining variables in* $\mathcal{H}$ *such that* $Ind_{\mathcal{G}}(X, Y, \mathbb{S})$ *holds. In the case when the appropriate subset* $\mathbb{S}$ *was found assign* $SepSet(X, Y) = \mathbb{S}$.
- *Initialize the set of v-structures* $\mathcal{C}$ *characterizing* $MC^{\mathcal{P}}_{\{T\}\cup MB_T^{\mathcal{P}}}$ *as an empty graph.*
- *For each triple of different vertices in* $\mathcal{H}$ (X, Z, Y) *such that the pairs* (X, Z) *and* (Y, Z) *are connected in* $\mathcal{H}$ *while the pair* (X, Y) *is not connected add* (X, Z, Y) *to the set* $\mathcal{C}$ *if* $Z \notin SepSet(X, Y)$.
- *Return the skeleton* $\mathcal{H}$ *and set of v-structures* $\mathcal{C}$ *characterizing* $MC^{\mathcal{P}}_{\{T\}\cup MB_T^{\mathcal{P}}}$.

The correctness of the above procedure, that is the ability to return the correct characterization of $MC^{\mathcal{P}}_{\{T\}\cup MB_T^{\mathcal{P}}}$ *assuming it exists, is a straightforward consequence of our previous analysis of correctness of the SGS and PC algorithm included in Section 4.2. The reason is that we apply here exactly the procedures appearing in these algorithms, with the only one difference that our source of the knowledge about d-separations occurring in the learned local structure is not derived from statistical tests for conditional independence, but from the knowledge about d-separations occurring in the whole graph* $\mathcal{G}$*—which is according to Remark 6.2.4.1 a correct source of knowledge.*

6.3.5. LBNA Example—Visualization

Figures 6.3.5.1, 6.3.5.2, 6.3.5.3, 6.3.5.4, 6.3.5.5, 6.3.5.6, 6.3.5.7 and 6.3.5.8 present all the inferred local faithful representations (skeletons equipped with v-structures) included in $\Lambda =$

$\{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$. *In each local structure the central node (this on the basis of which the Markov blanket subset supporting local structure is built) is drawn in a bold circle. Each occurring v-structure is denoted appropriately on the skeleton. For example, in Figure 6.3.5.3 there appears one v-structure:* (T, E, L). *All these representations have been obtained with the assist of the above described procedure.*

It is easy to check that each of these representations expresses indeed correctly some Markov equivalence class—all of them can be completely directed resulting in a DAG containing no additional v-structures, and moreover each such DAG contains exactly the same d-separations as the global graph $\mathcal{G}$ *among the corresponding to this local structure subset of its nodes. So according to Remark 6.2.4.1 the Markov equivalence classes of faithful representations* $\Lambda = \{MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X} : X \in \mathcal{V}\}$ *indeed here exist, and the are represented in the aforementioned figures.*

The LBNA theoretical version, that is Algorithm 6.3.1.1, is visualized in Figure 6.3.5.9. As we can see LBNA clearly leads to obtaining the correct Markov equivalence class of perfect maps of the whole distribution $\mathcal{P}$. *Let us investigate it a bit.*

After performing Step 2 of Algorithm 6.3.1.1 we obtain, as it should happen, the correct skeleton, as all the neighbors of X *in the skeleton characterizing* $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ *for each* $X \in \mathcal{V}$ *appears also in the skeleton characterizing* $MC^{\mathcal{P}}_{\mathcal{V}}$. *Step 3 is optional, especially in the considered here theoretical conditions, where we assume that on the entry we have perfectly represented local structures. And as it was expected it does not change anything.*

The dashed lines in the local structures in Figure 6.3.5.9 correspond to these edges which do not occur in the desired global skeleton, but are included in these local ones. The source of the LBNA methodology, in particular the recipe for the global skeleton construction included in Part (a) of Proposition 6.2.1.1, assures that these edges are excluded—namely the second condition assures this, as each of the additional edges does not occur in some other local structures containing the corresponding pairs of vertices. Namely: the edge (E, S) *does not occur in* $MC^{\mathcal{P}}_{\{L\}\cup MB^{\mathcal{P}}_L}$, (B, L) *does not occur in* $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$, *and* (B, E) *does not occur both in* $MC^{\mathcal{P}}_{\{B\}\cup MB^{\mathcal{P}}_B}$ *and* $MC^{\mathcal{P}}_{\{E\}\cup MB^{\mathcal{P}}_E}$.

In Step 5 there are two different v-structures appearing in local structures: (T, E, L) *and* (E, D, B)*—and they both are consistent with the skeleton obtained in previous steps, so they both will be included to the characterization of* $MC^{\mathcal{P}}_{\mathcal{V}}$.

As we can see, the illustrated on the right side of Figure 6.3.5.9 resultant model represents exactly the desired Markov equivalence class of graphs determined by the desired graph $\mathcal{G}$ *from Figure 6.3.4.1.*

Figure 6.3.5.1: $MC^{\mathcal{P}}_{\{A\}\cup MB^{\mathcal{P}}_A}$

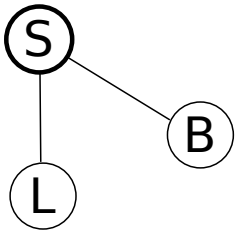


Figure 6.3.5.2: $MC^{\mathcal{P}}_{\{S\}\cup MB^{\mathcal{P}}_S}$

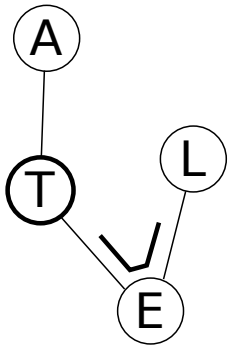


Figure 6.3.5.3: $MC^{\mathcal{P}}_{\{T\}\cup MB^{\mathcal{P}}_T}$

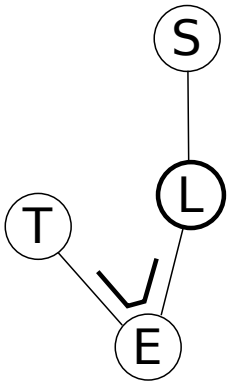


Figure 6.3.5.4: $MC^{\mathcal{P}}_{\{L\}\cup MB^{\mathcal{P}}_L}$

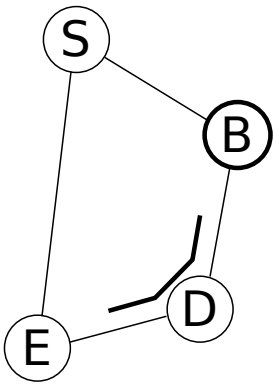


Figure 6.3.5.5: $MC^{\mathcal{P}}_{\{B\}\cup MB^{\mathcal{P}}_B}$

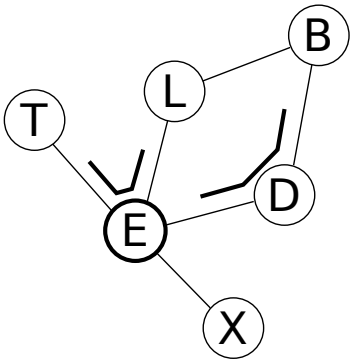


Figure 6.3.5.6: $MC^{\mathcal{P}}_{\{E\}\cup MB^{\mathcal{P}}_E}$

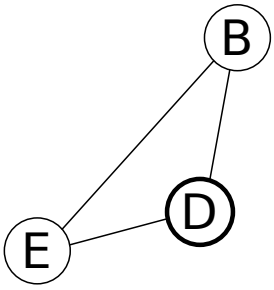


Figure 6.3.5.7: $MC^{\mathcal{P}}_{\{D\}\cup MB^{\mathcal{P}}_D}$

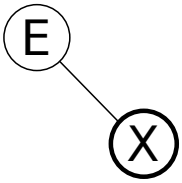


Figure 6.3.5.8: $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$

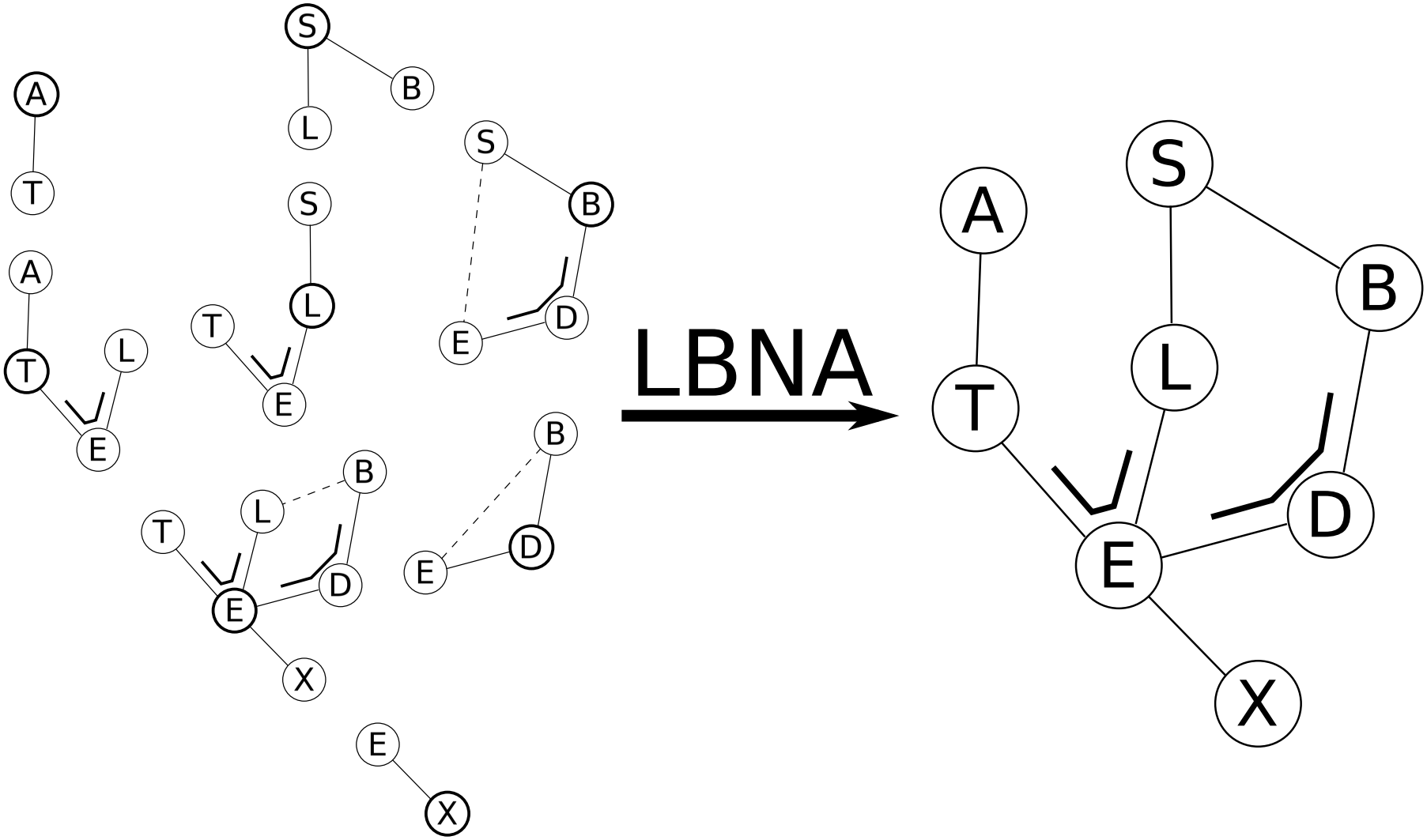


Figure 6.3.5.9: LBNA visualization—retrieving the Asia Bayesian network structure from local models.

6.4. LBNA—Practical Version

In this section we introduce a practical realization of the considered in the previous section theoretical version of the LBNA methodology. The first subsection explains briefly what are the main differences occurring here. The remaining subsections discuss in details these differences.

6.4.1. Differences Comparing to Theoretical Version

There are two main differences here with respect to the considered in the previous section scenario:

- We do not assume here any a priori given perfect knowledge about the local faithful representations. Instead we consider here the practical situation where as usually on the entry we have some dataset $\mathcal{D} = (U, V)$ available. Also as usually we treat such table $\mathcal{D}$ as a finite sample of some unknown joint probability distribution $\mathcal{P}$ of the set of attributes $\mathcal{V}$. So what we first have to do is to learn the local structures on the basis of this dataset.

- After learning the local structures the previously described theoretical version of LBNA—Algorithm 6.3.1.1 can be applied in order to convert them into some representation of the global model. However, we must be aware that as the input we have here potentially wrong local models. And as a consequence the result of merging them into the global model can be also wrong.

  In particular, the obtained characterization: the skeleton and set of v-structures, can be inconsistent, similarly as in the number of described in previous chapters constraint-based algorithms. So we need to adopt here some repairing procedures in order to convert the potentially inconsistent model obtained with Algorithm 6.3.1.1 into the correct representation of some Markov equivalence class.

6.4.2. Auxiliary Subroutines

The practical version of the LBNA algorithm, which we apply on $\mathcal{D}$ in order to learn a corresponding to it Bayesian network structure, depends on two parameters:

- $\mathcal{MC}$—a Bayesian network structure learning algorithm, which calculates on the basis of $\mathcal{D}$ for a given nonempty subset of attributes $\mathbb{X} \subseteq \mathcal{V}$ the approximation $\widehat{MC^{\mathcal{P}}_{\mathbb{X}}}$ of the Markov equivalence class of all perfect maps of $\mathcal{P}\,|_{\mathbb{X}}$, that is the approximation of $MC^{\mathcal{P}}_{\mathbb{X}}$—assuming that this local faithful representation exists. As we have seen it in previous chapters such assumption is typical for the constraint-based algorithms—and it of course cannot even be verified without any further knowledge, that is having only the table $\mathcal{D}$—which is just a finite sample. This assumption is most of all important with respect to the investigation of the theoretical soundness of such algorithms. Exactly analogical situation is here. That is, what is required in practice from an algorithm $\mathcal{MC}$ is to return a presumably good Bayesian network structure—the skeleton and the set of v-structures included in this structure are then treated as a representation of $\widehat{MC^{\mathcal{P}}_{\mathbb{X}}}$. Only in theoretical conditions, like in Proposition 6.2.1.1, where we assume that appropriate local perfect maps exist and we exactly know them, merging these local networks into a global perfect structure is possible. What we hope when applying this proposition in a practical implementation is that high quality local Bayesian networks can be merged using the introduced recipe into a high quality global model.

- $\mathcal{MB}$—an algorithm calculating on the basis of $\mathcal{D}$ for a given attribute $X \in \mathcal{V}$ the approximation $\widehat{MB_X^{\mathcal{P}}}$ of the Markov blanket of X with regard to the distribution $\mathcal{P}$, that is the approximation of $MB_X^{\mathcal{P}}$. This should be understood in exactly the same way as in the case of $\mathcal{MC}$. Without any knowledge whether the distribution $\mathcal{P}$, from which the sample $\mathcal{D}$ is generated, admits a faithful representation—we even do not know whether Markov blankets are unique. But again, we simply do not take care about all the unverifiable assumptions, on the basis of which the theoretical soundness of the approach is guaranteed by Proposition 6.2.1.1. What is in practice just required from $\mathcal{MB}$ is to return a subset of attributes considered as a strong candidate for a Markov blanket, assuming it is unique. Alternatively $\mathcal{MB}$ can be a Bayesian network structure learning algorithm, The reason is that our aim is to determine $\widehat{MB_X^{\mathcal{P}}}$ for all $X \in \mathcal{V}$, and these approximations can be obtained from a network structure, if we apply the recipe included in Theorem 4.3.1.1.

Consequently, the resultant global model obtained from the practical LBNA procedure we will denote by $\widehat{MC_{\mathcal{V}}^{\mathcal{P}}}$, in order to emphasize that in theoretical conditions, given all required assumptions stated in Proposition 6.2.1.1, what is returned by the procedure of merging is a perfect model $MC_{\mathcal{V}}^{\mathcal{P}}$.

6.4.3. High Level Description

LBNA, which we denote from now as LBNA($\mathcal{MC}$, $\mathcal{MB}$), to point out its characterizing parameters—auxiliary algorithms, can be described at a high level of abstraction quite simply—see Algorithm 6.4.3.1.

Algorithm 6.4.3.1 LBNA($\mathcal{MC}$, $\mathcal{MB}$)—the practical version.

---

1. Apply the algorithm $\mathcal{MB}$ to the table $\mathcal{D}$ in order to determine $\widehat{MB_X^{\mathcal{P}}}$ for each $X \in \mathcal{V}$.
2. Apply the algorithm $\mathcal{MC}$ to the table $\mathcal{D}$ in order to determine $\widehat{MC^{\mathcal{P}}_{\{X\} \cup \widehat{MB_X^{\mathcal{P}}}}}$ for each $X \in \mathcal{V}$.
3. Use Algorithm 6.3.1.1 with optional Step 3 and with the given on the input local models $\widehat{MC^{\mathcal{P}}_{\{X\} \cup \widehat{MB_X^{\mathcal{P}}}}}$ for all $X \in \mathcal{V}$ in order to determine the skeleton $\mathcal{G}$ and set of v-structures $\mathcal{C}$ characterizing the global model.

   Additionally, for each v-structure included in $\mathcal{C}$ during the run of Algorithm 6.3.1.1 store its frequency of being caught to this set, that is in how many local models it appears.
4. Apply the repairing procedure in order to convert the potentially inconsistent representation $(\mathcal{G}, \mathcal{C})$ of some Markov equivalence class into the correct one—with the assist of stored frequencies in local models of each v-structure included in $\mathcal{C}$.

The first two points of the procedure does not need any additional comments—the important aspects of applying $\mathcal{MC}$ and $\mathcal{MB}$ have been already explained. The applied in Step 3 Algorithm 6.3.1.1 we have also explained in details in the previous section—however here the additional counter for each v-structure expressing its frequency in the processed local models is created. This counter is used in the last repairing step, which requires more attention.

We must precise what is in fact the result of the whole practical LBNA procedure. The merging procedure in Step 3 creates the Markov equivalence class representation—the skeleton and set of v-structures, which are potentially inconsistent, and require further processing. The repairing procedure in Step 4 creates instead a directed acyclic graph, representing a global Bayesian network structure. From this DAG in particular a representation of $\widehat{MC}_{\mathcal{V}}^{\mathcal{P}}$ in terms of a skeleton and a set of v-structures can be directly determined. The inspiration for our repairing procedure was the similar approach included in the MMHC method (Algorithm 5.3.5.1).

6.4.4. Resolving Directions

The reason why we need Step 4 lies in the problem with the procedure of v-structures induction, which in practice is insufficient to be applied in the form given in Proposition 6.2.1.1. As it has been previously mentioned, although in theoretical conditions determined by the assumptions of Proposition 6.2.1.1 everything is fine and a perfect global model can be deduced from perfect local structures—in practice these assumptions do not hold and are even not verifiable. In particular after performing Step 2 the obtained models $\widehat{MC}^{\mathcal{P}}_{\{X\}\cup\widehat{MB}_X^{\mathcal{P}}}$ for $X \in \mathcal{V}$ are simply imperfect. This can cause huge problems when applying directly the recipe for v-structures induction, that is Part (b) of the proposition—which happens in the case of applied in Step 3 Algorithm 6.3.1.1.

The original recipe can be characterized as an inclusion of all suitable v-structures appearing in local models. By suitable v-structures we mean here these which are consistent with an already obtained from Part (a) global skeleton, that is v-structures consisting of edges included in undirected versions in our global skeleton. When local models are imperfect, then the acquired from them set of v-structures describing a global model can be not only imperfect—it can and typically will be even not a possible set of v-structures of any DAG based on the already learned skeleton.

In other words we face here the typical problem occurring in the number of described earlier constraint-based Bayesian network structure learning algorithms, where as the output, or a partial result, the characterization of the Markov equivalence class—the skeleton and set of v-structures—is obtained. Also here some v-structures can lead to a contradiction about directions of some arcs (some two v-structures can determine direction of the same arc in opposite way), or some subset of these v-structures can determine a cycle in a network, or the obtained set of v-structures can be insufficient in the sense that every DAG based on the learned skeleton and consistent with this set of v-structures have at least one more v-structure.

In order to bypass all the above mentioned problems, but in the same time stick at least partially to the theoretically correct recipe in Part (b) of the proposition, we adopt in LBNA a greedy approach to convert the learned with respect to the proposition skeleton and set of v-structures in Step 3 into a concrete DAG. Namely, the procedure is presented in Algorithm 6.4.4.1.

First of all we greedily reduce the potentially inconsistent set of v-structures $\mathcal{C}$ obtained in Step 3 of Algorithm 6.4.3.1, corresponding to the direct application of Proposition 6.2.1.1, to the most promising subset of v-structures which at least does not lead to any cycles and bidirectional edges (although it still can be inconsistent in the sense that any DAG consistent with the skeleton $\mathcal{G}$ and updated here set of v-structures $\mathcal{C}$ has always at least one more not occurring in $\mathcal{C}$ v-structure). We perform the reduction of the set $\mathcal{C}$ with respect to the v-structures counter obtained earlier—in order to prevent from exclusion the most frequently

Algorithm 6.4.4.1 Details of Step 4 of Algorithm 6.4.3.1.

Let $\mathcal{G}$ and $\mathcal{C}$ be respectively the learned in Step 3 of Algorithm 6.4.3.1 skeleton and set of v-structures.

1. Initiate the partially directed graph $\mathcal{K}$ as a completely undirected graph equal to the skeleton $\mathcal{G}$.
2. Order the v-structures from the set $\mathcal{C}$ from the most to the least frequent–according to the created in Step 3 of Algorithm 6.4.3.1 counter. Proceed each candidate according to this order, and direct corresponding to it edges in $\mathcal{H}$ in directions consistent with these in the v-structure if no arc with two opposite directions and no cycle appears in $\mathcal{H}$—otherwise do not insert this v-structure to $\mathcal{H}$ and remove it from $\mathcal{C}$.
3. Let us initialize the DAG $\mathcal{H}$ as an empty graph operating on the set of nodes $\mathcal{V}$.
4. Repeat:
   - Determine the DAG $\mathcal{H}'$ which is the neighbor of $\mathcal{H}$ maximizing value $Score(\mathcal{D}, \mathcal{H}')$. The neighbor of $\mathcal{H}$ means here any DAG which can be obtained from $\mathcal{H}$ after applying one of the operations: removing, reversing or adding one edge—provided that we do not remove or reverse any edge occurring in some v-structure belonging to the set $\mathcal{C}$, that no cycle appears in the case of reversion or addition, and that any added edge exists in undirected version in $\mathcal{G}$.
   - If $Score(\mathcal{D}, \mathcal{H}') > Score(\mathcal{D}, \mathcal{H})$ assign $\mathcal{H} := \mathcal{H}'$, otherwise terminate the loop.
5. Return the DAG $\mathcal{H}$ as a resultant Bayesian network structure.

appearing in local models v-structures.

Then as it can be noticed we apply a very similar to the one occurring in the MMHC algorithm Hill-Climbing procedure, in order to completely resolve directions in the learned structure. In the case of MMHC the only restriction of the search in the space of DAGs corresponds to not exceeding the previously learned skeleton. Here some additional restrictions appear—namely we are limited only to the space of DAGs which has fixed directions of all edges corresponding to the set of v-structures $\mathcal{C}$ (fixed and consistent with these v-structures). Similarly as in the case of MMHC the *Score* function is an arbitrary chosen by a user scoring criterion. According to this scoring criterion we search in Step 4 of Algorithm 6.4.4.1 for the valuable structure satisfying the restrictions determined by the learned skeleton $\mathcal{G}$ and set of v-structures $\mathcal{C}$.

6.5. LBNA Performance

One of the reasons why the LBNA procedure has been proposed was a hope that such approach basing on presumably small local and thus quite accurately learned with some common algorithm $\mathcal{MC}$ local networks can lead us to a more accurately learned global model than in the case of directly learning it with the method $\mathcal{MC}$. Another hope was that LBNA can in the same time achieve in practice at least comparable time scalability to the global learning approach. How these hopes were meaningful in the face of reality will be the main topic of the next experiments chapter. This chapter we end with some more detailed explanation of the second hope.

Namely, in this subsection we investigate the performance of LBNA methodology with respect to several cases. We start with a more theoretical case, but finish with a purely practical one, in particular corresponding exactly to the scenario which we phase in reality—reported in details in the next chapter.

6.5.1. Analysis Scheme

We have already analyzed the time complexity of the theoretical version of LBNA in two scenarios—without any assumptions, and with the assumption that the input local structures have bounded size, that is each of them consists of at most k nodes for some common constant bound k.

The practical version of LBNA (Algorithm 6.4.3.1) contains in particular this theoretical part (Step 3), but it also includes the initial process of local structures learning from the input dataset (Step 1 and 2), and the final repairing procedure of the potentially inconsistent obtained in Step 3 global model (Step 4). First of all we will analyze with respect to the above mentioned two scenarios the part of Algorithm 6.4.3.1 corresponding to Step 3 and 4—that is without taking into consideration the local models learning routine. Similarly as previously we perform this analysis only with respect to the time complexity aspect, as no statistical tests for conditional independence are applied here. We split the analysis however into two cases corresponding to assumptions regarding the lack or the presence of the theoretical correctness of local models obtained in Steps 1 and 2—as these assumptions have the influence on the efficiency of Step 4.

As in the case of MMHC we exclude here from the analysis the performance of the final greedy Hill-Climbing search. Pessimistically this search is exponential in time with regard to n—the number of attributes. Pessimistically it travels through the exponentially many structures until it reaches some local maximum—regardless of the simplifying assumptions of limited space of the search space. In practice however we did not observe this step as a time bottleneck in the whole LBNA procedure, thus we include here also the time complexity analysis of the remaining parts of this method.

It is clear that the performance in any aspect of the initial process of local structures learning depends directly on the used auxiliary Markov blanket learning algorithm $\mathcal{MB}$ and Bayesian network structure learning algorithm $\mathcal{MC}$. We will perform the separate analysis for these steps in the case of particular choice of $\mathcal{MB}$ and $\mathcal{MC}$—the choice applied in all described in Chapter 7 experiments with LBNA. The chosen method in both roles is MMHC. As there are in particular performed statistical tests for conditional independence in this method we extend here our analysis to all scenarios and aspects applied in the case of constraint-based methods in all this work, defined in Section 4.1.

6.5.2. No Auxiliary Subroutines Practical Case

In the case when we do not assume anything about the local models merged in Step 3 of Algorithm 6.4.3.1 we can summarize the time complexity of this algorithm with regard to n—the number of nodes in the learned global model, as follows:

- Step 3 costs us $\mathcal{O}(n^4)$ time—as we have already found this bound in Subsection 6.3.3 for the executed here Algorithm 6.3.1.1 (the additional creation of v-structures counter does not change this pessimistic bound).
- In Step 4 (that is in Algorithm 6.4.4.1) we pessimistically have to process $\mathcal{O}(n^4)$ v-structures (as pessimistically we can have $\mathcal{O}(n^3)$ v-structures in each of the n local

structures). We first sort them according to the counter, which costs us $\mathcal{O}(n^4)\log(n^4)$ time. Then we process in this order each of these v-structures and try to insert it to the the updated partially directed graph $\mathcal{H}$, checking in particular whether no cycle appears in $\mathcal{H}$. Each such cycle detection in $\mathcal{H}$ costs us pessimistically $\mathcal{O}(n^2)$ time if we apply the depth-first search algorithm (as graph $\mathcal{H}$ pessimistically contains $\mathcal{O}(n^2)$ directed edges). So overall we have that the process of the reduction of the set $\mathcal{C}$ costs us $\mathcal{O}(n^6)$ time.

Summarizing, the time complexity of Steps 3 and 4 of Algorithm 6.4.3.1 excluding the final Hill-Climbing search is $\mathcal{O}(n^6)$.

In the case when we assume that the local models merged in Step 3 of Algorithm 6.4.3.1 have the bounded number of nodes by k we have that the time complexity of creating the global model with regard to n and k is as follows:

- Step 3 costs us $\mathcal{O}(nk^3)$ time—according to the conclusions reached in Subsection 6.3.3 (again the cost of the additional counter creation is included in the overall cost of the theoretical version of LBNA).

- In Step 4 we pessimistically process here only $\mathcal{O}(nk^3)$ v-structures—as there are included in the previous step to the set $\mathcal{C}$ at most k^3 v-structures from each of the n local models. Sorting this set of v-structures according to the counter costs us $\mathcal{O}(nk^3\log(nk^3))$ time. Note that in the updated DAG $\mathcal{H}$ with the processed in the obtained order v-structures there can appear pessimistically only $\mathcal{O}(nk)$ directed edges—as all the $\mathcal{O}(nk^3)$ processed v-structures, each of them resulting in at most two new directed edges in $\mathcal{H}$, are included in the global learned skeleton in Step 3, which clearly has at most $\mathcal{O}(nk)$ edges. As a consequence checking whether $\mathcal{H}$ contains cycles costs pessimistically $\mathcal{O}(nk)$ time. We perform this check for each of the processed v-structures, so overall we have that the process of the reduction of the set $\mathcal{C}$ costs us $\mathcal{O}(n^2k^4)$ time.

We have obtained that the time complexity of Steps 3 and 4 of Algorithm 6.4.3.1 excluding the Hill-Climbing part is in the case of limited by k processed local structures equal to $\mathcal{O}(n^2k^4)$.

6.5.3. No Auxiliary Subroutines Theoretical Case

Let us now assume a purely theoretical setting where we have on the entry of Step 3 perfect local models—the local faithful representations basing on real Markov blankets in some global joint probability distribution $\mathcal{P}$ admitting a faithful representation. In such case the above described time complexity in the case of local structures of unlimited size remains the same. However some reduction of this pessimistic complexity can be observed in the case of local structures of size limited by the common bound k.

Namely, let us notice that in this case not only the global learned skeleton in Step 3 of Algorithm 6.4.3.1 have at most $\mathcal{O}(nk)$ edges, but also for sure each node has at most k neighbors in this skeleton. The reason is that in the considered here theoretical setting the set of neighbors of any variable X in the corresponding to it local structure $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$ corresponds exactly to the set of neighbors of X in the global model $MC^{\mathcal{P}}_{\mathcal{V}}$ (no more neighbors in the global model can appear, as any neighbor of X in particular belongs to $MB^{\mathcal{P}}_X$—so it must be connected with X in $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$).

The consequence of the bounded by k number of neighbors of each node in the global skeleton is that in Step 3 there can be gathered no more than $\mathcal{O}(nk^2)$ different v-structures consistent with this global skeleton and derived from local models (for each node X at most

k^2 v-structures with the center node corresponding to X can appear). The cost of sorting them is $\mathcal{O}(nk^2 \log(nk^2))$, and for each of them checking that the updated graph $\mathcal{H}$ is acyclic have still time cost $\mathcal{O}(nk)$. As the result the time cost of the reduction of the set $\mathcal{C}$ in Step 4 of Algorithm 6.4.3.1 in the considered here theoretical case is reduced to $\mathcal{O}(n^2k^3)$, and as the result the total time cost of Steps 3 and 4 excluding the final Hill-Climbing part is reduced to $\mathcal{O}(n^2k^3)$.

Summarizing, we have obtained that both in the theoretical and practical case the time complexity of the LBNA merging and repairing mechanism excluding Hill-Climbing is in the case of limited local structures quadratic with regard to n. Although the pure theoretical version of LBNA has in this case linear with regard to n time complexity this is still a very good bound. Let us remind that typically at least quadratic with regard to n time cost has already the initial process in Step 1 of the Markov blanket learning of each attribute.

6.5.4. MMHC Subroutine Case—Motivation

Now let us perform a typical for the previous chapters complex analysis of the whole Bayesian network structure learning method LBNA($\mathcal{MC}$, $\mathcal{MB}$) given in Algorithm 6.4.3.1. On the entry we have a dataset $\mathcal{D} = (\mathcal{U}, \mathcal{V})$ containing m objects and n attributes. As in previous chapters, the whole analysis is performed with regard to these two dimensions: m and n.

As we have already mentioned it, we use in this analysis in the role of both the $\mathcal{MC}$ and $\mathcal{MB}$ the MMHC algorithm. The application of MMHC in the role of $\mathcal{MB}$ in Step 1 of Algorithm 6.4.3.1 means learning the global, operating on all attributes, Bayesian network structure, and then extracting the Markov blanket of each attribute directly from this structure with an assist of the characterization given in Theorem 4.3.1.1.

In order to empirically evaluate the LBNA mechanism we have employed exactly the above setting—LBNA(MMHC, MMHC). That is why we perform here the analysis with regard to such version. There are two main reasons why we have decided to test such choice:

- MMHC is one of the most advanced and successful in practice Bayesian network structure learning methods in the history of such algorithms. Regardless of the application—the local structures learning subroutine $\mathcal{MC}$, or the Markov blankets learning subroutine $\mathcal{MB}$— it is a reasonable and efficient choice.

- We will compare in Chapter 7 in the reported experiments the LBNA(MMHC, MMHC) approach in particular with the MMHC algorithm itself. It is very interesting to see how the arrangement of such strong algorithm like MMHC in the LBNA scheme can change the quality and efficiency of the learning, comparing to the direct application of MMHC.

What we will perform here is the comparison of these methods from the theoretical point of view defined in Section 4.1, in terms of pessimistic bounds expressing efficiency and reliability of both methods. In the case of the time complexity aspect we do not include into this analysis the cost of the final Hill-Climbing search. In fact whichever further considered scenario we take, in each of them including the cost of this search would increase the pessimistic complexity bounds of both LBNA(MMHC, MMHC) and MMHC to the exponential with regard to n (in both methods MMHC is invoked globally on all attributes—so in both cases this search is conducted for such global structures).

6.5.5. MMHC Subroutine Case—Scenario ST_1, ST_2 and SP_1

We start with scenario ST_1, where we assume only that there exists a generative distribution $\mathcal{P}$ of the input sample $\mathcal{D}$ admitting a faithful representation.

Let us analyze first the pessimistic number of performed statistical tests for conditional independence. In Step 1 of Algorithm 6.4.3.1 we apply the algorithm MMHC in order to learn the global Bayesian network structure expressing the distribution $\mathcal{P}$. According to Subsection 5.2.10 and 5.3.3 we pessimistically perform here the exponential with regard to n number of tests. So in particular the MMHC method itself similarly as LBNA(MMHC, MMHC) performs here pessimistically such number of tests.

The pessimistic size of the conditional set in performed tests is in the case of pure MMHC equal here to $n-2$—again according to Subsection 5.2.10 and 5.3.3. This is the largest possible size, so also in LBNA(MMHC, MMHC) this pessimistic size is equal to $n-2$—as MMHC is there in particular globally invoked.

The time complexity of LBNA(MMHC, MMHC) can be summarized as follows:

- In Step 1 of Algorithm 6.4.3.1 the MMHC method is invoked globally, and according to Subsection 5.3.3 this algorithm has pessimistically exponential with regard to n time complexity, and linear with regard to m.
- The remaining steps in Algorithm 6.4.3.1 does not increase this pessimistic cost.

So also in the case of the time complexity both approaches LBNA(MMHC, MMHC) and MMHC are equally scalable.

Nothing changes in the scenario ST_2—where we assume the existence of the faithful representation of the generative distribution having limited degree of each node, and in the scenario SP_1—where we do not assume anything. According to Subsections 5.2.10, 5.2.11 and 5.3.3 MMHC performance in all aspects remains here unchanged comparing to the scenario ST_1. These pessimistic bounds with regard to the number of performed tests, pessimistic size of the conditional set in these tests and time complexity achieved in Step 1 of Algorithm 6.4.3.1, where MMHC is invoked globally, are not exceeded in the remaining steps of this algorithm. So in all these scenarios MMHC and LBNA(MMHC) are pessimistically equally effective in all aspects.

6.5.6. MMHC Subroutine Case—Scenario SP_2

As usually the most interesting scenario is the one most common in practice—where the only one assumed thing is that we apply the Spirtes et al. (2000) convention of neglecting the least reliable statistical tests for conditional independence. This is in particular the scenario strictly applied in the presented in Chapter 7 experiments. Let us remind that here in the case of binary attributes of the input dataset the largest possible conditional set for which the test is normally conducted is equal to $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$, where c is a user-specified constant—and in the case of attributes having more values the bound for the largest possible conditional set is for sure not bigger.

We apply here as a subroutine the MMHC algorithm—and only in this subroutine some tests for conditional independence are calculated. So in the case of neglected tests we automatically assume a conditional independence—which is the natural strategy in the case of MMHC. In particular we have already discussed some performance aspects of MMHC in the scenario SP_2 with such strategy of neglecting tests—so we will further simply refer to the obtained there results.

Let us recall that in the case of the scenario SP_2 the largest possible size of the resultant set of neighbors of a target node returned by the MMHC subroutine—the MMPC algorithm—is equal to $b_m + 1$. The additional "symmetry correction" applied in MMHC can potentially reduce this set—but pessimistically its size can be still equal to $b_m + 1$. The skeleton obtained with the assist of MMPC is an upper bound for the final learned skeleton in MMHC—so in this resultant skeleton each node has degree at most $b_m + 1$. The immediate consequence of this is that in the Bayesian network structure returned by MMHC the extracted from it Markov blanket of each node according to the characterization given in Theorem 4.3.1.1 has size bounded by $(b_m + 1)^2$.

Keeping in mind the above fact let us analyze the performance of LBNA(MMHC, MMHC) in the scenario SP_2. We start with the pessimistic number of performed tests for conditional independence:

- In Step 1 of Algorithm 6.4.3.1 we invoke globally on the whole dataset $\mathcal{D}$ MMHC—and according to Subsection 5.2.12 and 5.3.3 we will perform pessimistically here $\mathcal{O}((m \log_2 m)n^2)$ tests.

- In Step 2 we learn with the assist of MMHC each of the n local structures. Each such structure is based on the subset of attributes corresponding to the learned in Step 1 Markov blanket of some attribute—including this attribute, so according to our previous observations each such local structure contains at most $(b_m + 1)^2 + 1$ nodes. Consequently, in order to learn each of these local structures we pessimistically perform $\mathcal{O}((m \log_2 m)b_m^4)$ tests, that is $\mathcal{O}(m(\log_2 m)^5)$ tests. So totally in Step 2 we pessimistically perform $\mathcal{O}(m(\log_2 m)^5 n)$ tests.

- In the remaining steps corresponding to the LBNA mechanism we do not perform any more tests for conditional independence.

So totally we have obtained that the pessimistic number of performed tests in LBNA(MMHC, MMHC) is $\mathcal{O}((m \log_2 m)n^2 + m(\log_2 m)^5 n)$. The pessimistic number of performed tests in the pure MMHC is equal to $\mathcal{O}((m \log_2 m)n^2)$. It is a better bound with respect to the number m of objects in $\mathcal{D}$, however both approaches are here equally well scalable with regard to the number n of attributes in $\mathcal{D}$.

The pessimistic size of the conditional set is according to Subsection 5.2.12 and 5.3.3 in the case of pure MMHC equal in this scenario to b_m. In LBNA(MMHC, MMHC) the same pessimistic size is achieved. Here the MMHC subroutine is not only invoked globally on all attributes, but also locally—however in each of these runs the direct reason of the limited by b_m size of the conditional sets in performed tests is the applied Spirtes et al. (2000) convention of neglecting less reliable tests.

The time complexity of LBNA(MMHC, MMHC) excluding the Hill-Climbing search in both the MMHC subroutine and the final step of LBNA itself can be summarized as follows:

- In Step 1 of Algorithm 6.4.3.1 the invoked globally auxiliary method MMHC costs $\mathcal{O}((m \log_2 m)^2 n^2)$ time.

- In Step 2 we learn each of the n local structures (each having at most $(b_m + 1)^2 + 1$ nodes) in $\mathcal{O}((m \log_2 m)^2 b_m^4)$ time, that is in $\mathcal{O}(m^2(\log_2 m)^6)$ time. Step 2 costs us then $\mathcal{O}(m^2(\log_2 m)^6 n)$ time.

- The cost of Step 3 and 4 of Algorithm 6.4.3.1 excluding Hill-Climbing we have already considered in the case of local structures of the size (number of nodes) limited by k—it

is according to Subsection 6.5.2 equal to $\mathcal{O}(n^2k^4)$. In our case this complexity is equal then to $\mathcal{O}(b_m^8 n^2)$, that is $\mathcal{O}((\log_2 m)^8 n^2)$.

Summarizing, after excluding all the Hill-Climbing search the time complexity of the pure MMHC algorithm is here $\mathcal{O}((m \log_2 m)^2 n^2)$, while the time complexity of LBNA(MMHC, MMHC) is $\mathcal{O}((m \log_2 m)^2 n^2 + m^2 (\log_2 m)^6 n + (\log_2 m)^8 n^2)$. So again, the pure MMHC method is better scalable with regard to m, however it is equally well scalable as the LBNA approach with regard to n.

The conclusion is that if we forget about the Hill-Climbing part, the time complexity with regard to the number of variables n of the LBNA(MMHC, MMHC) is in fact the same as the complexity of the MMHC method itself—which is invoked in LBNA in Step 1. The final Hill-Climbing search can be potentially some bottleneck in the case of both approaches, however in the experiments described by Tsamardinos et al. (2006) such greedy procedure was tested (with the purpose of its application in MMHC, that is without v-structures limitations, only with the skeleton restrictions)—and it did not cause any serious problems. In fact the MMHC algorithm was very well time scalable comparing to all other approaches analyzed in this work. It is hard to draw any conclusions how the additional v-structures restriction appearing in the Hill-Climbing part of the LBNA approach affects its time performance. We will however clearly see the comparison between these two strategies achieved in practice in Chapter 7.

6.5.7. General Case Discussion

Let us finish this section with the following brief analysis concerning the more general case of any auxiliary algorithms used in the role of $\mathcal{MC}$ and $\mathcal{MB}$ in the LBNA mechanism.

The main strength of the LBNA($\mathcal{MC}$, $\mathcal{MB}$) time efficiency lies in the aspect of the scalability of the approach with regard to the number of attributes n of the input dataset, in the case of the scenario SP_2, that is when the Spirtes et al. (2000) convention is applied:

- In such case most of the classical constraint-based Markov blanket learning approaches used in the role of $\mathcal{MB}$, including these considered in this work, returns Markov blankets of limited by some constant size, but in the same time the process of learning all these blankets, that is Step 1 of Algorithm 6.4.3.1, have at least quadratic with regard to n time complexity.

- Consequently in Step 2 of Algorithm 6.4.3.1 all the learned models are based on the subsets of attributes of some bounded size—so whatever Bayesian network structure learning algorithm we use, each such local model will be learned in $\mathcal{O}(1)$ time, and as the result Step 2 will have the time complexity $\mathcal{O}(n)$.

- As we already know Steps 3 and 4 corresponding to the the LBNA merging mechanism excluding the final Hill-Climbing search costs in the case of limited local models $\mathcal{O}(n^2)$ time.

Summarizing, the LBNA time cost with regard to n can be expected here to be equal or even better than the first its step, corresponding to learning the blankets. However, we must remember that some risk in the total efficiency of the LBNA method is always related to its final greedy part.

Let us assume that we want to compare for some fixed choice of algorithms $\mathcal{MC}$ and $\mathcal{MB}$ the time efficiency of the following two Bayesian network learning approaches: simply applying the method $\mathcal{MC}$ on the whole dataset, and applying LBNA($\mathcal{MC}$, $\mathcal{MB}$)—assuming

again that the scenario SP_2 is considered. It can be expected that for many choices of the $\mathcal{MC}$ and $\mathcal{MB}$ methods the time complexity of applying $\mathcal{MC}$ on the whole dataset will be at least as big as the time complexity of applying $\mathcal{MB}$ in order to learn the Markov blanket of each attribute. The intuition about this can be that learning a global Bayesian network model gives de facto some approximations of all Markov blankets, if we use the blanket characterization given in Theorem 4.3.1.1 or 4.3.1.2. So in terms of the time efficiency LBNA($\mathcal{MC}$, $\mathcal{MB}$) can potentially be the comparable choice to the direct usage of $\mathcal{MC}$, or even better. We have exactly seen this in the case of $\mathcal{MC}$ set to MMHC.

It must be pointed out here that in general there is no guarantee that obtained during the LBNA procedure Markov blankets have indeed limited size—if we for example do not use the Spirtes et al. (2000) practice for neglecting the least reliable statistical tests, or if we do not use here any constraint-based solution. In such cases, in particular in many times considered earlier scenarios ST_1, ST_2 and SP_1, as well as any other not considered in this dissertation scenario one can imagine, where we do not control anyhow the pessimistic running time of the local structures learning subroutine, the pessimistic time complexity of the whole LBNA Bayesian network structure learning method can be exponential with regard to n. However, the intuition is that somehow the above nice properties of LBNA in the scenario SP_2 can hold in some other scenarios, if we work with a specific datasets—for example samples generated from a joint distribution, in which the Markov blanket of each attribute has indeed limited by some fixed constant size. Still there is no guarantee that obtained by LBNA blankets approximations will also have this property, but at least to some extent such characterization of the input should positively affect the time performance of LBNA.

Chapter 7

Local Bayesian Networks Aggregation—Empirical Evaluation

This chapter provides a detailed overview of the relatively large number of performed experiments with the LBNA approach. The main aim of these experiments is to analyze the practical usefulness of the whole proposed here framework—in several various aspects, and with two other recognizable and very strong in specific applications Bayesian network structure learning algorithms. One of them, MMHC, has been discussed widely in the previous chapters. This algorithm is also applied in the LBNA mechanism itself, as we have mentioned it earlier. The second one, GES, was only mentioned so far—we will say something more about it here.

After some general notes about the common features of all the conducted tests we present in this chapter a detailed description of the obtained results, together with the interpretation and analysis of them. Most of the resultant statistics have been moved to Appendix A in order to increase the readability of this part of work.

7.1. Experiments Description

The aim of this section is to express in details what kind of experiments have been performed. We in particular list here the compared algorithms, including LBNA, expressing in details their setting and the implementation source. We introduce all the benchmark Bayesian networks on the basis of which the training datasets are derived, and with regard to which the quality analysis of compared methods is done. We describe all applied quality measures, expressing various aspects of the methods goodness. Finally we describe the main experiment scheme, on the basis of which all empirical comparison was performed.

7.1.1. Compared Methods—Description

All described in this section experiments have been implemented in and performed using the *R* environment for statistical computing (R Development Core Team, 2014) with an additional help of the command line version of the TETRAD software (see `http://www.phil.cmu.edu/projects/tetrad/`). Most extensively the *R* package *bnlearn* (Scutari, 2010, 2011; Nagarajan et al., 2013) was applied. In particular the implementation of the LBNA method itself is written in R, with an assist of this package. In Section 7.3 we present all details necessary to use these tools and perform described further experiments.

In the experiments we have compared the following three Bayesian network learning methods:

- MMHC—Max-Min Hill-Climbing (Tsamardinos et al., 2006)—the described in details in Section 5.3 hybrid (partially constraint-based and partially score-based) method, with a very sophisticated constraint-based mechanism. This is a well time scalable and in the same time accurate in comparison to many other state-of-the-art algorithms Bayesian network structure learning approach. We have used implementation of the MMHC approach included in the *bnlearn* package. All the parameters of this method (they can be seen on page `http://www.bnlearn.com/documentation/man/hybrid.html`) were set to default except:

  - the scoring criterion used in the Hill-Climbing part—which was for the purpose of these experiments set to the BDeu score with the default equivalent sample size: 10 (Heckerman et al., 1995).
  - the statistical test for conditional independence used in the MMPC part—which was here set to the G^2 test with adjusted degrees of freedom df_{adj}^T (we have described in details such test in Section 2.2, in particular in Subsection 2.2.7). A default significance level was applied, that is 0.05. Each test in the *bnlearn* MMHC implementation is performed according to the Spirtes et al. (2000) convention (described in Subsection 2.2.8)—only provided that there are at least 5 samples for each cell in the conditional probability table corresponding to the test—otherwise the conditional independence is automatically assumed.

  The purpose of the above setting was to test the MMHC version fully compatible with the original MMHC implementation reported by Tsamardinos et al. (2006). Indeed, in the experiments described in this work exactly the above chosen scoring criterion in the Hill-Climbing search was applied, as well as exactly the same methodology of testing conditional independencies.

- GES—Greedy Equivalent Search (Chickering, 2002)—a very recognizable score-based Bayesian network learning method. This is an asymptotically optimal greedy procedure, in the limit of large sample leading to a faithful representation whenever it exists. This algorithm can potentially return results of outstanding quality. It was for example the only one algorithm which was able to beat the MMHC approach in terms of finding a closer to reality model in some of the performed experiments reported by Tsamardinos et al. (2006). The main problem however in the case of the GES method seems to be its poor time scalability with regard to the increasing number of variables, resulting from the pessimistically exponential branching factor in the performed search. We have used the GES implementation included in the previously mentioned version of the TETRAD software. In particular we have used here the same scoring criterion as in the case of MMHC—the BDeu score with the equivalent sample size 10.

- LBNA(MMHC, MMHC)—Local Bayesian Networks Aggregation—the proposed in this work approach, namely the practical version of this method specified in Algorithm 6.4.3.1 and 6.4.4.1, where in the role of both the auxiliary algorithms $\mathcal{MB}$ and $\mathcal{MC}$ the MMHC method was used. This MMHC is in particular the same *bnlearn* implementation with the same parameters setting as the one described in the first point of this list. Let us recall that in the case of $\mathcal{MB}$ applying the MMHC method means simply learning the global, operating on all the attributes of the considered dataset, Bayesian network

with this method, and then obtaining all the Markov blankets automatically using the recipe included in Theorem 4.3.1.1. Similarly as in the used MMHC method in the tested LBNA version we have employed in the final Hill-Climbing search part the BDeu score with the equivalent sample size 10.

7.1.2. Compared Methods—Returned Models Unification

We will further need the following two definitions.

Definition 7.1.2.1 (Compelled edge) *For the given Markov equivalence class $\mathcal{C}$ by the compelled edge we will understand such directed edge which occurs in the given direction in every DAG which is the member of $\mathcal{C}$.*

Definition 7.1.2.2 (Completed PDAG) *For the given Markov equivalence class $\mathcal{C}$ the completed PDAG is a partially directed graph which has the skeleton exactly the same as the one representing $\mathcal{C}$, and each of its edge is directed if and only if the two connected vertices correspond to some compelled edge in $\mathcal{C}$—and in such case it has the same direction as this compelled edge.*

The skeleton and set of v-structures are, according to Theorem 2.4.3.1, a sufficient and unique representation of any Markov equivalence class. A completed PDAG is a kind of complement of this representation, giving some extra information about all fixed directed edges for the considered class—that is about all compelled edges.

The original MMHC algorithm (Tsamardinos et al., 2006) returns already a fully directed acyclic graph—in particular the same does the *bnlearn* implementation. The GES implementation which we have used returns however a completed PDAG representing one Markov equivalence class. This result we further process in the *R* environment with an assist of the *bnlearn* package, in order to obtain from this partially directed graph a completely directed acyclic graph which is an exemplary representative of the corresponding Markov equivalence class. As we have explained it in Subsection 6.4.4 the LBNA method, similarly as the MMHC approach, returns already a completely directed acyclic graph.

In fact for most of the gathered during further described experiments statistics it is enough to work with results in the form of Markov equivalence classes—so for example with completed PDAGs. The only exceptions are in fact statistics corresponding to some kind of prediction using learned models, where we work with complete Bayesian networks, that is DAGs with learned parameters. In the case of all three tested here Bayesian network structure learning algorithms the final parameters learning in any obtained structure is done with an assist of Maximum Likelihood parameters estimation.

7.1.3. Training Datasets

All the presented further experiments have been performed using datasets generated from real-world expert Bayesian networks. These networks have been designed by domain experts. They are aimed at several tasks like prediction, decision making, diagnoses, estimation or troubleshooting. Some of them have become an internal and key element of the recognizable expert systems. Table 7.1.3.1 summarizes all these networks from the point of view of such characteristics (the Munin1, Munin2, Munin3 and Munin4 networks are the subnetworks of the whole Munin model, which is listed in this table). Table 7.1.3.2 gives some additional technical informations about each of the 18 considered here Bayesian networks. The networks are ordered by the number of contained variables.

Name	Description	Authors
Child	The diagnostic system for telephone referrals of newborn babies with possible congenital heart disease.	(Spiegelhalter and Cowell, 1992)
Insurance	A network for estimating the expected claim costs for a car insurance policyholder.	(Binder et al., 1997)
Water	A model of the biological processes of a water purification plant.	(Jensen et al., 1989)
Mildew	A model for deciding on the amount of fungicides to be used against attack of mildew in wheat.	(Jensen and Jensen, 1996)
Alarm	A medical expert network for monitoring patients in intensive care.	(Beinlich et al., 1989)
Barley	A decision support system for growing malting barley without use of pesticides.	(Kristensen and Rasmussen, 2002)
Hailfinder	A system forecasting severe summer hail in northeastern Colorado.	(Abramson et al., 1996)
Hepar II	A medical expert network for the diagnosis of liver disorders.	(Oniśko, 2003)
Win95pts	An expert system for printer troubleshooting in Windows 95.	`http://research.microsoft.com/`
Pathfinder	An expert system assisting surgical pathologists with the diagnosis of lymph-node diseases.	(Heckerman et al., 1992)
Andes	The student modeling component of an intelligent tutoring system for Newtonian physics.	(Conati et al., 1997)
Diabetes	A differential equation-based Bayesian network model for insulin dose adjustment.	(Andreassen et al., 1991)
Pigs	A pedigree of breeding pigs used for diagnosing the PSE disease.	`http://cs.au.dk/~csj/`
Link	A model for the linkage between two genes: human LQT syndrome and a genetic marker.	(Jensen and Kong, 1996)
Munin	An expert electromyography assistant diagnosing neuromuscular diseases (consists of four subnetworks).	(Andreassen et al., 1989)

Table 7.1.3.1: The table summarizes for each of the considered networks their practical purpose. An interested in more details reader can find here also some further informations about the authors of each network, most often with some direct references to appropriate articles.

Name	Nodes	Edges	Parameters	Variables domain size	Markov blankets size
Child	20	25	230	2/3(1.17)/6	1/3(2.152)/8
Insurance	27	52	1008	2/3.296(0.993)/5	1/5.185(2.543)/10
Water	32	66	10083	3/3.625(0.492)/4	1/7.688(3.623)/13
Mildew	35	46	540150	3/17.6(27.01)/100	1/4.571(2.048)/9
Alarm	37	46	509	2/2.838(0.727)/4	1/3.514(2.077)/8
Barley	48	84	114005	2/8.771(9.047)/67	2/5.25(2.81)/13
Hailfinder	56	66	2656	2/3.982(1.721)/11	1/3.536(2.703)/17
Hepar II	70	123	1453	2/2.314(0.627)/4	1/4.514(5.633)/26
Win95pts	76	112	574	2/2(0)/2	1/5.921(4.569)/29
Pathfinder	109	195	72079	2/4.110(5.909)/63	1/3.817(10.341)/107
Munin1	186	273	15622	2/5.333(3.582)/21	1/3.806(2.58)/17
Andes	223	338	1157	2/2(0)/2	0/5.614(3.758)/23
Diabetes	413	602	429409	3/11.337(5.882)/21	2/3.966(2.651)/48
Pigs	441	592	5618	3/3(0)/3	2/3.655(4.098)/68
Link	724	1125	14211	2/2.532(0.825)/4	0/4.801(4.397)/31
Munin2	1003	1244	69431	2/5.36(3.667)/21	1/3.314(2.293)/30
Munin4	1038	1388	80352	2/5.438(3.638)/21	1/3.532(3.336)/70
Munin3	1041	1306	71059	2/5.38(3.665)/21	1/3.333(3.326)/70

Table 7.1.3.2: Several statistics describing applied in the experiments expert Bayesian networks, namely: the name of the network, number of nodes, edges and parameters in the network, the minimal/average(standard deviation)/maximal size of variables domain and size of Markov blankets. All decimal values are rounded to three digits.

All the networks have been downloaded from the web page `http://www.bnlearn.com/bnrepository/`. Some more statistics of each network can be found there.

These networks have, as we can see, very different characteristics. However some common features can be noted here. When we look at the number of edges in comparison to the number of variables we can agree that all these networks are quite sparse. Except the Water network there are never more than two edges for one vertex on average, and in the case of biggest networks this ratio is in fact much closer to 1 than 2. Secondly, although the maximal Markov blankets sizes occurring in the networks does not look like they are limited by some constant—the average and standard deviation of these sizes indicate that we most often deal with really small and bounded blankets (it must be pointed out however, that some of the blankets outliers are really huge—containing almost all the network, as in the case of the Pathfinder graph). This is a good information for us, as it seems that such characteristic should strengthen the scalability and accuracy of the LBNA algorithm.

For the given Bayesian network we generate datasets of some declared number of rows corresponding to this network, that is consisting of the same attributes. Each created dataset is a sample generated from the joint probability distribution represented by this network (see Definition 2.1.3.1). The sample generation from the network is done using the approach exactly the same as the one applied in the described in Subsection 3.1.3 logic sampling algorithm.

7.1.4. Measures of Quality—Introduction

At least in the case when parameters of the given DAG are chosen randomly in order to construct some Bayesian network the theoretical results given by Meek (1995) guarantee in

particular an important property of the joint probability distribution represented by such network. Namely, almost surely (with a probability equal to 1) the joint probability distribution represented by the constructed network admits a faithful representation—and the structure of this network is an exemplary perfect map of this distribution. This means that we a priori know for such Bayesian network what is the perfect model describing the corresponding to it joint probability distribution. Thanks to this we have an easy way to measure a quality of Bayesian network learning algorithms—as we in advance know how a perfect Markov equivalence class representation should look for a distribution from which each dataset is generated.

In our case we work with the a priori constructed Bayesian networks, where the parameters setting is not just the result of some random choice. However, we still can expect that the structures of these expert networks approximately correspond to the perfect maps of represented by them joint probability distributions. Thus our evaluation of obtained models by the GES, MMHC and LBNA methods is with regard to the structures of these expert networks used to generate the input datasets for our algorithms.

We will now explain in details step by step all the statistics used in order to evaluate the considered approaches.

Assume that we work with an artificially generated dataset obtained as a sample from the distribution held by one of the expert Bayesian networks—let us name this network $\mathcal{BN}_{real}$. Then let us consider that we apply on this dataset some Bayesian network structure learning algorithm, which returns some network structure from which a Bayesian network $\mathcal{BN}_{learned}$ is obtained. Most of the following introduced here measures aim at expressing from different sides the quality of the learned model $\mathcal{BN}_{learned}$ with respect to the real model $\mathcal{BN}_{real}$.

7.1.5. Structure Closeness-Based Measures of Quality

In order to evaluate the applied learning method we can compare how close to the theoretically perfect model $\mathcal{BN}_{real}$ (which structure is treated as an exemplary perfect map) is the learned network $\mathcal{BN}_{learned}$. For the sake of simplicity for any Bayesian network $\mathcal{BN}$ let us introduce the following notation:

- $Sk(\mathcal{BN})$—the set of all undirected edges occurring in the skeleton of the network $\mathcal{BN}$,
- $Vs(\mathcal{BN})$—the set of all v-structures appearing in the structure of the network $\mathcal{BN}$.

Skeleton/V-structures Precision/Recall

In all further described experiments in particular the following closeness criteria are used:

- skeleton precision—a real value between 0 and 1, expressing how many edges occurring in $\mathcal{BN}_{learned}$ are correct in the sense that they also appear in $\mathcal{BN}_{real}$:

$$\frac{|Sk(\mathcal{BN}_{real}) \cap Sk(\mathcal{BN}_{learned})|}{|Sk(\mathcal{BN}_{learned})|},$$

- skeleton recall—a real value between 0 and 1, expressing how many edges occurring in $\mathcal{BN}_{real}$ are detected by the learning algorithm in the sense that they also appear in $\mathcal{BN}_{learned}$:

$$\frac{|Sk(\mathcal{BN}_{real}) \cap Sk(\mathcal{BN}_{learned})|}{|Sk(\mathcal{BN}_{real})|},$$

- v-structures precision—a real value between 0 and 1, expressing how many v-structures occurring in $\mathcal{BN}_{learned}$ are correct in the sense that they also appear in $\mathcal{BN}_{real}$:

$$\frac{|Vs(\mathcal{BN}_{real}) \cap Vs(\mathcal{BN}_{learned})|}{|Vs(\mathcal{BN}_{learned})|},$$

- v-structures recall—a real value between 0 and 1, expressing how many v-structures occurring in $\mathcal{BN}_{real}$ are detected by the learning algorithm in the sense that they also appear in $\mathcal{BN}_{learned}$:

$$\frac{|Vs(\mathcal{BN}_{real}) \cap Vs(\mathcal{BN}_{learned})|}{|Vs(\mathcal{BN}_{real})|}.$$

The above measures clearly aim at expressing how the learned model is close to the perfect Markov equivalence class for the joint probability distribution being a sample generator, by examining the two main characterizations of every Markov equivalence class: a skeleton and a set of v-structures. The appearing here distinction of the measures with respect to the precision and recall of the skeleton and v-structures will be later helpful in order to get better understanding of the experiments results.

Structural Hamming Distance

But we have also used one more measure expressing the similarity of equivalence classes corresponding to the learned and real model. The measure is called Structural Hamming Distance (SHD) and has been proposed by Tsamardinos et al. (2006). It compares in fact the completed PDAGs representing respectively the learned and real model. It is simply equal to the number of elementary operations leading from the one completed PDAG to the second, where by an elementary operation we understand adding, removing, reversing, directing or undirecting one edge. The SHD metric gives us a kind of summarizing information indicating how similar are the learned and real equivalence classes. We will treat it further as the major indicator showing how close to the reality models each of compared methods can find. Some more detailed information in aspects of separately the skeleton and v-structures quality the previously introduced measures will give.

Normalized Number of Edges

The next measure we will call a normalized number of edges. This is a very simple indicator, showing only how dense is the learned model comparing to the real one. It is equal to the number of edges occurring in the network $\mathcal{BN}_{learned}$ divided by the number of edges occurring in the network $\mathcal{BN}_{real}$. If this ratio is more than 1 then the learned model is denser than the real one, while the value less than 1 means that we deal with the sparser learned model.

7.1.6. Remaining Quality Measures

In addition to the measures directly aimed at showing the closeness of the learned and real models we have applied some other ones, which give an insight into some different aspects. Let us list now all these remaining measures, reflecting completely different aspects.

Normalized Score

The first measure we will call a normalized score. By $Score(\mathcal{D}, \mathcal{BN})$ for some given dataset $\mathcal{D}$ and Bayesian network $\mathcal{BN}$ consisting of the same attributes as the table we will understand

here the BDeu score with equivalent sample size 10 calculated for $\mathcal{BN}$ with respect to $\mathcal{D}$. The normalized score of some learned Bayesian network $\mathcal{BN}_{learned}$ on the basis of the generated from the true network $\mathcal{BN}_{real}$ dataset $\mathcal{D}$ we will understand as $\frac{Score(\mathcal{D},\mathcal{BN}_{learned})}{Score(\mathcal{D},\mathcal{BN}_{real})}$. The score is always negative (as it is a logarithm of some positive value less than 1, so the lower is the normalized score for some learned model $\mathcal{BN}_{learned}$ the higher BDeu score corresponds to this network. In particular the normalized score equal to 1 means that both learned an real models are equally strong in the sense of the BDeu score calculated on the generated dataset, while the normalized score less than 1 means that the learned network has even a higher BDeu score than the true network.

It must be pointed out that the lower normalized score does not necessarily means the better model in terms of the similarity to the real graph. Asymptotically when the sample size grows the BDeu score would of course be the clear indicator awarding the best networks, but in the presented further experiments we deal only with some small in fact samples—especially small in the case of biggest expert networks in the role of sample generators. So from this measure we can rather get an information which of the compared methods aims at finding higher scored graphs—regardless of the true meaningfulness of such founded models.

Prediction Accuracy

The another applied measure is a prediction accuracy. This measure tries to evaluate how good is the learned on the basis of the dataset generated from some expert network model in terms of using it as a classifier for some newly generated from the same expert network samples. In order to calculate this measure we generate from the considered expert network not only the training dataset on which some networks are learned, but also the second, test dataset—on which the prediction accuracy of the learned model is estimated. The prediction is performed with respect to the randomly chosen decision for each next row of the test dataset. That is, for each next row: we randomly and independently choose one variable which will represent a decision attribute, and then we perform the prediction of the value of this attribute given the knowledge of the values of all remaining attributes in this row (we choose the most probable value of this attribute according to the values of the remaining attributes and the learned network—and in the case of two or more equally and top probable values we choose the first one founded most probable value). The prediction accuracy of the learned Bayesian network is then equal to the number of rows in the test sample correctly classified using this network divided by the number of all rows in this sample.

This measure gives an insight into the important aspect of the quality of the learned model—but it is again not so obviously correlated with the similarity of this model to the real training and test datasets generator, as it will be clearly seen in the experiments results. We will further discuss this topic meanwhile presenting these results.

Execution Time

We have applied one more still not mentioned measure—an execution time. For the given method applied on some generated dataset it is simply equal to the elapsed time in seconds corresponding to performing the network learning. This is the only one measure not related to the quality of the obtained results by each of compared methods. But it is a very important aspect of the practical usefulness of these algorithms.

7.1.7. Experiments Scheme

For each expert Bayesian network $\mathcal{BN}$ from the set of 18 considered in these experiments and introduced in Section 7.1.3 and for each $s \in \{128, 512, 2048, 8192\}$ we have performed the following procedure:

1. For each compared Bayesian network structure learning algorithm $\mathcal{M}$:
   - We have randomly generated from the network $\mathcal{BN}$ 128 samples of size s. That is we have obtained as the result 128 datasets $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{128}\}$ with attributes corresponding to the network $\mathcal{BN}$ and each of them consisting of s rows. These are our training datasets.
   - We have additionally randomly generated from the network $\mathcal{BN}$ the next 128 datasets $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{128}\}$, where this time each of them consists of 1024 rows. These are our test datasets.
   - For each $i \in \{1, 2, ..., 128\}$ we have performed Bayesian network learning using the algorithm $\mathcal{M}$ on the basis of dataset $\mathcal{D}_i$, and then we have found the value of each of described measures corresponding to the network returned by the method $\mathcal{M}$, the table $\mathcal{D}_i$ and the true expert model $\mathcal{BN}$, that is: SHD, skeleton precision and recall, v-structures precision and recall, normalized number of edges, normalized score, prediction accuracy and execution time. The prediction accuracy for the network learned on the training dataset $\mathcal{D}_i$ was calculated on the test dataset $\mathcal{T}_i$.
   - For each considered quality measure we have calculated the average and standard deviation of this measure results obtained for the method $\mathcal{M}$ over our 128 training datasets.
2. For some of the pairs among considered methods we have statistically compared both methods with regard to each considered quality measure. In order to do it for each of the nine measures we have calculated the two-sided permutation test (Good, 2000) with 2^{20} random permutations basing on the obtained for both algorithms 128 measure values corresponding to the 128 generated training datasets. The null hypothesis of each such test is that both compared methods perform the same according to the considered measure. The significance level was set to 5%—that is in the case when the p-value corresponding to some conducted test is less than 0.05 we claim that the difference between two compared methods with respect to the considered measure is significant—and as the significant winner we treat this one which has achieved better average value of the measure over 128 generated training datasets (better might mean smaller or bigger, depending on the measure, see description in Appendix A). Otherwise we claim that there is no significant winner in this case.

We further go into details of all obtained using the above procedure results. The remaining part of this chapter divides and order these results—with the purpose of giving a clear view and interpretation of them.

7.2. Results Analysis

Appendix A presents a detailed information about the obtained results in terms of quality measures after conducting the above described experiment for all 18 datasets and each of the 4 sample sizes—comparing the three described in Subsection 7.1.1 methods: GES, MMHC and LBNA(MMHC, MMHC). It is recommended for the reader to go now into the appendix and

read first of all the initial description, where we have explained precisely how to understand all the resultant tables appearing there.

From now we assume that the reader has already studied the initial description appearing in the appendix. Similarly as in this appendix, we will further write for simplicity everywhere in this subsection LBNA instead of LBNA(MMHC, MMHC). In the further part we collect some clear conclusions derived from all the results appearing there.

7.2.1. Significant Winners—Visualization

In order to better visualize these results we have converted the information about the significant winners included in all the tables into the compact graphical form. The nine figures from 7.2.1.1 to 7.2.1.9 present the information about the significant winners for respectively each of the nine considered quality measures—as it can be seen in the captions. Each of the figures consists of two tables. The higher table shows graphically which method is a significant winner with regard to the corresponding to the figure measure among the three compared methods. Each cell of the table, representing one experiment for the given expert network and sample size, has one of the five possible colors with the following interpretations:

- Red—the significant winner in the case of the indicated experiment and quality measure is the GES method.
- Green—the significant winner is here the MMHC method.
- Blue—the significant winner is here the LBNA method.
- White—there is no any significant winner here.
- Gray—in the case of the GES algorithm the memory limit was exceeded, and as the result the comparison between all three approaches has not been achieved. Each of the three compared methods was run with the memory usage limited to 6 GB. Only in the case of GES there has appeared a situation where this limit was exceeded—especially in the case of the few biggest expert networks.

The lower table shows which method is a significant winner with regard to the corresponding measure among only the two algorithms: MMHC and LBNA. Both these methods have managed to perform the learning in the limit of 6 GB of memory for the case of all the samples generators—expert networks, and for all the sample sizes. Thus here in each cell, again corresponding to one experiment for the given expert network and sample size, we have only three possible colors, with the analogical to the previous table interpretations:

- Green—the significant winner is for the corresponding experiment and quality measure the MMHC method.
- Blue—the significant winner is here the LBNA method.
- White— there is no any significant winner here.

The expert networks are ordered in all these tables according to the number of contained variables.

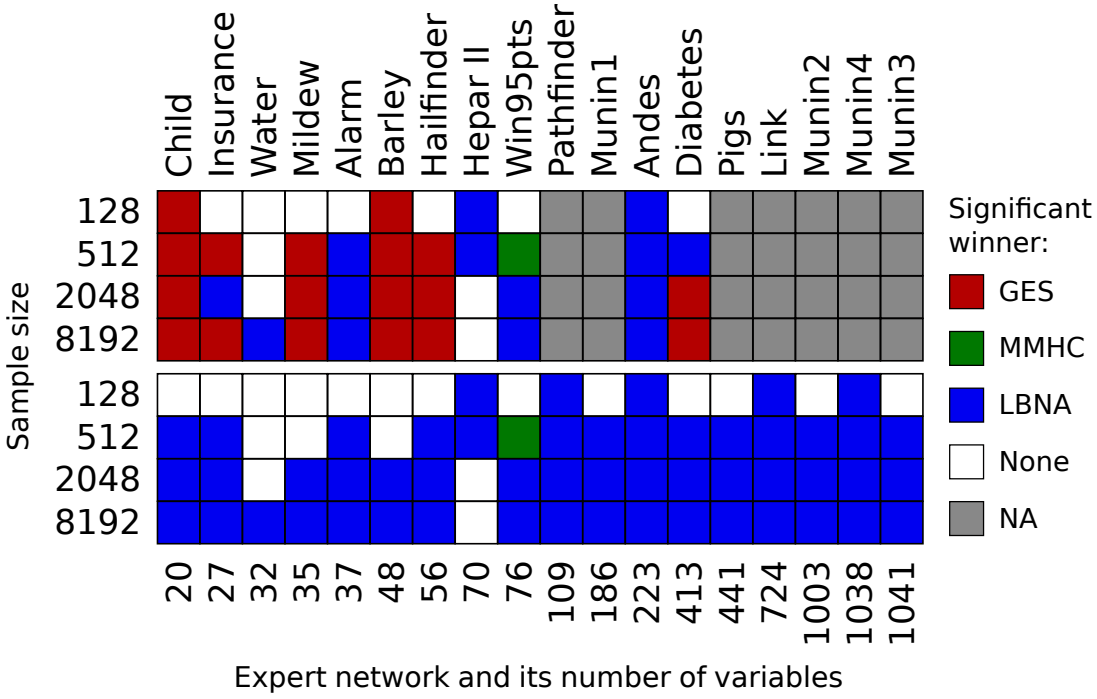


Figure 7.2.1.1: Structural Hamming Distance

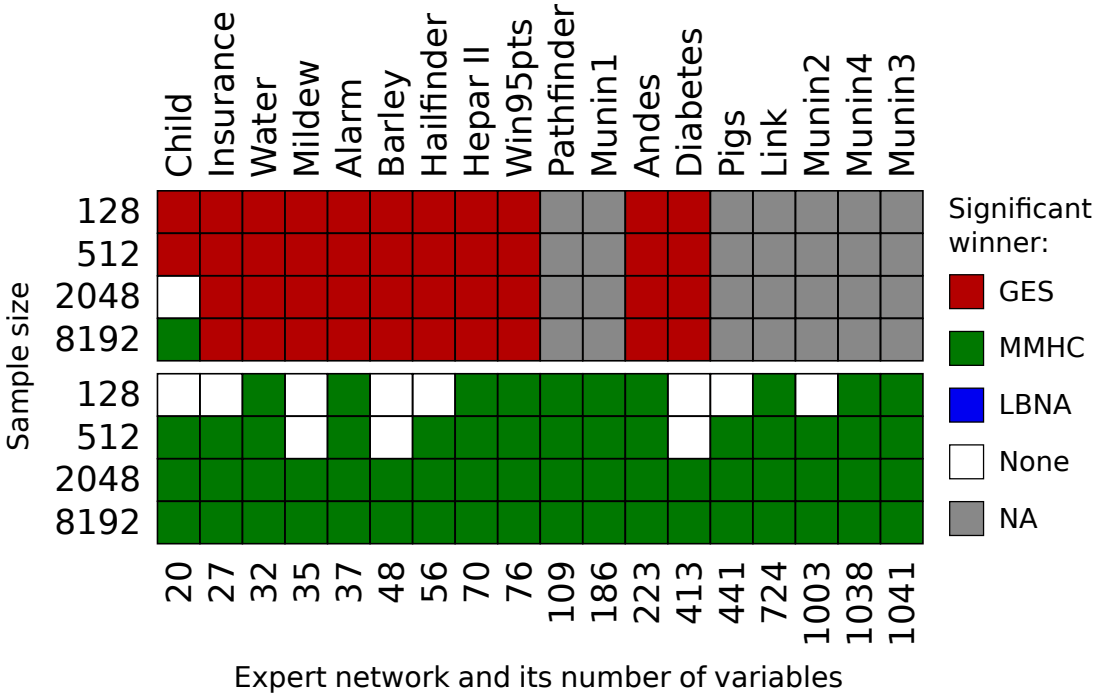


Figure 7.2.1.2: Normalized number of edges

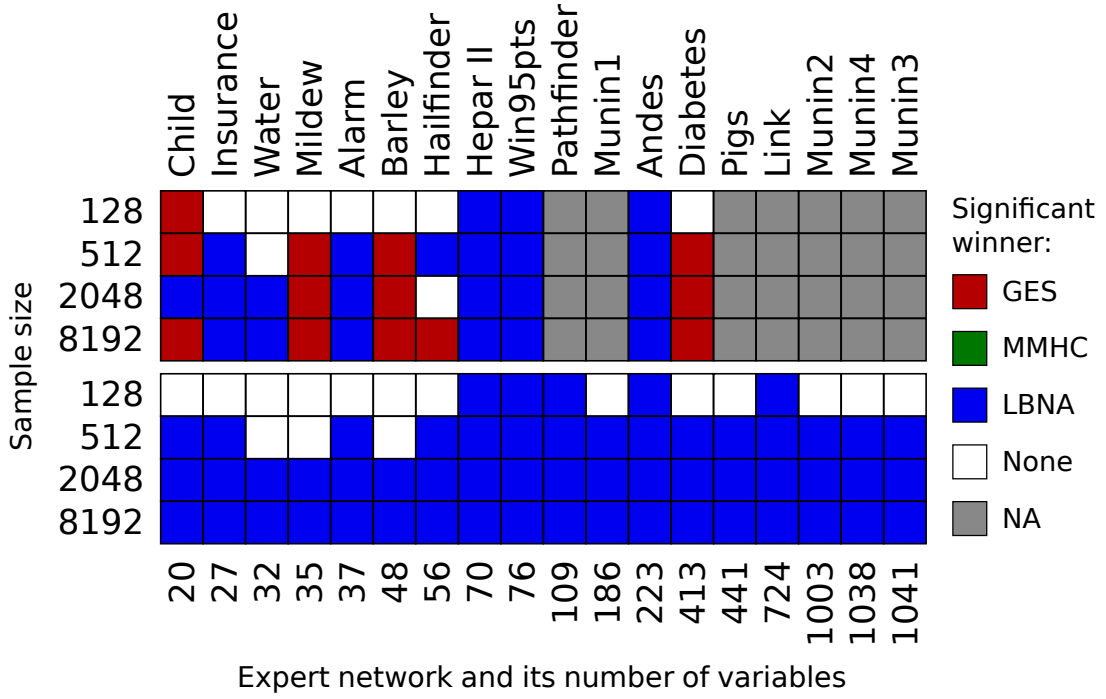


Figure 7.2.1.3: Skeleton precision

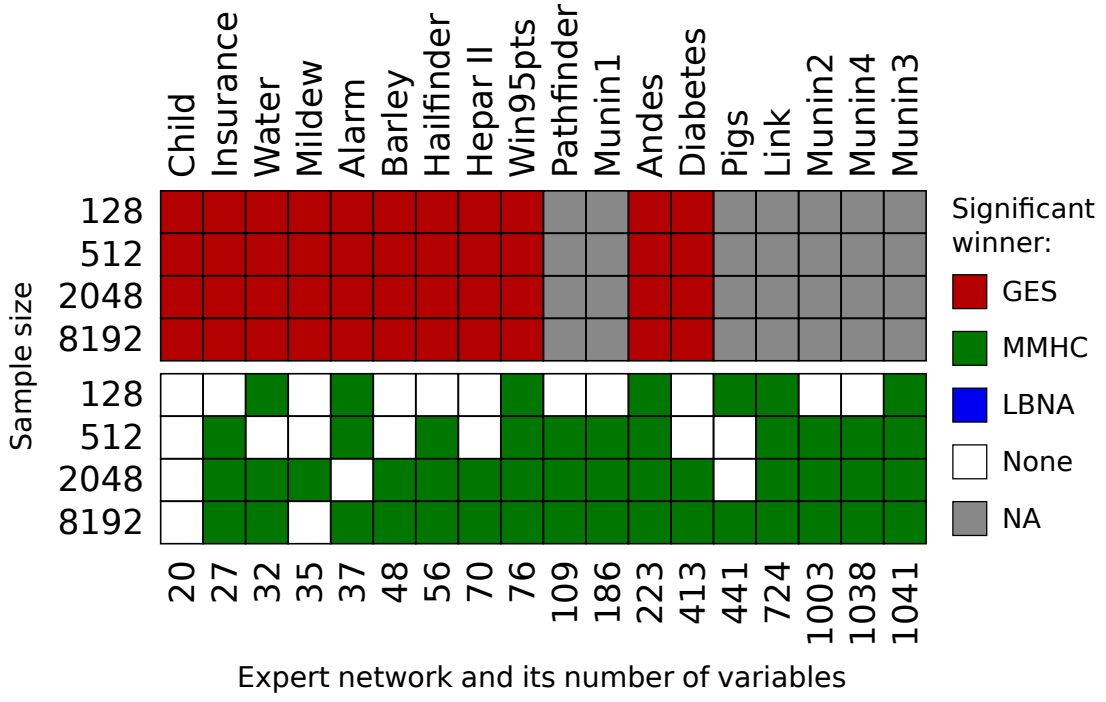


Figure 7.2.1.4: Skeleton recall

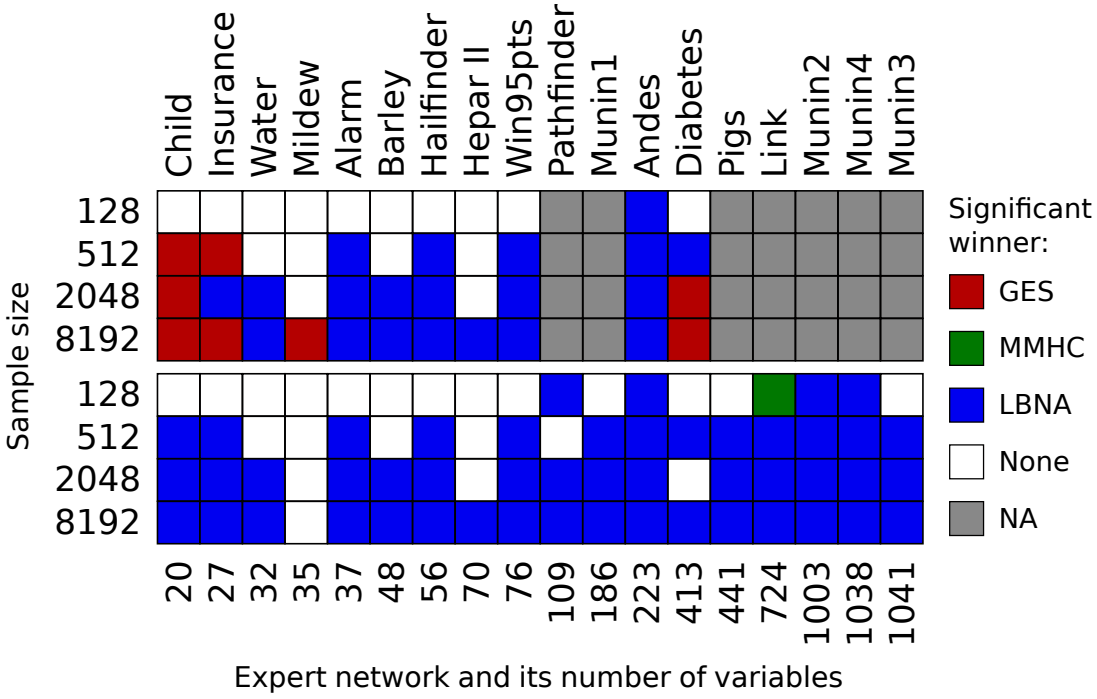


Figure 7.2.1.5: V-structures precision

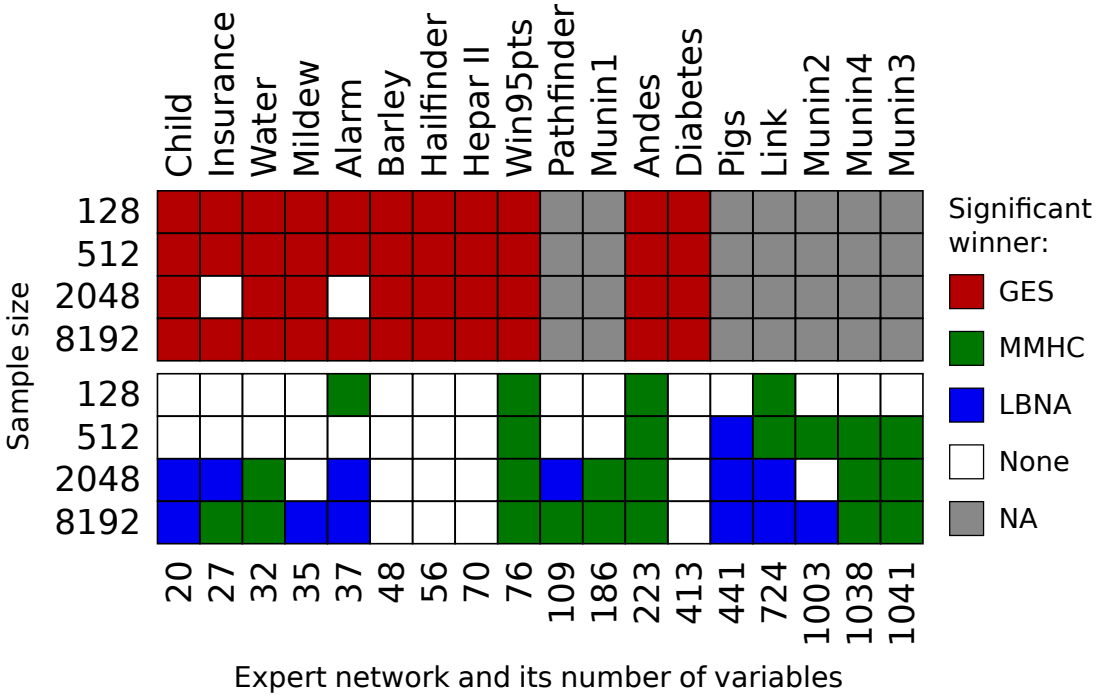


Figure 7.2.1.6: V-structures recall

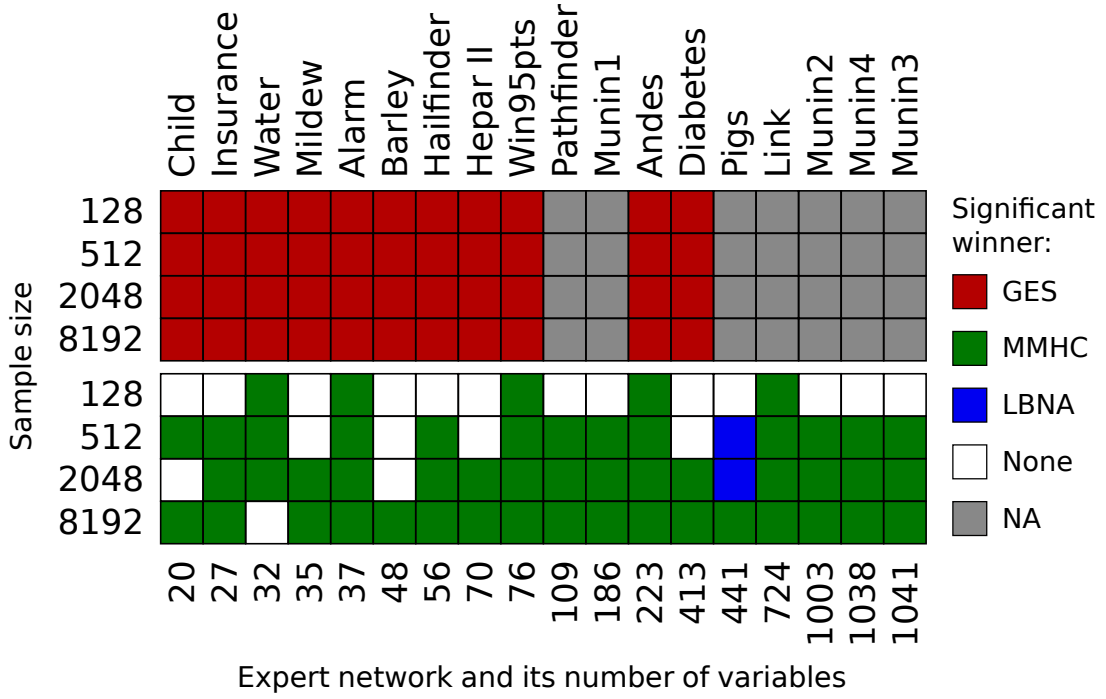


Figure 7.2.1.7: Normalized score

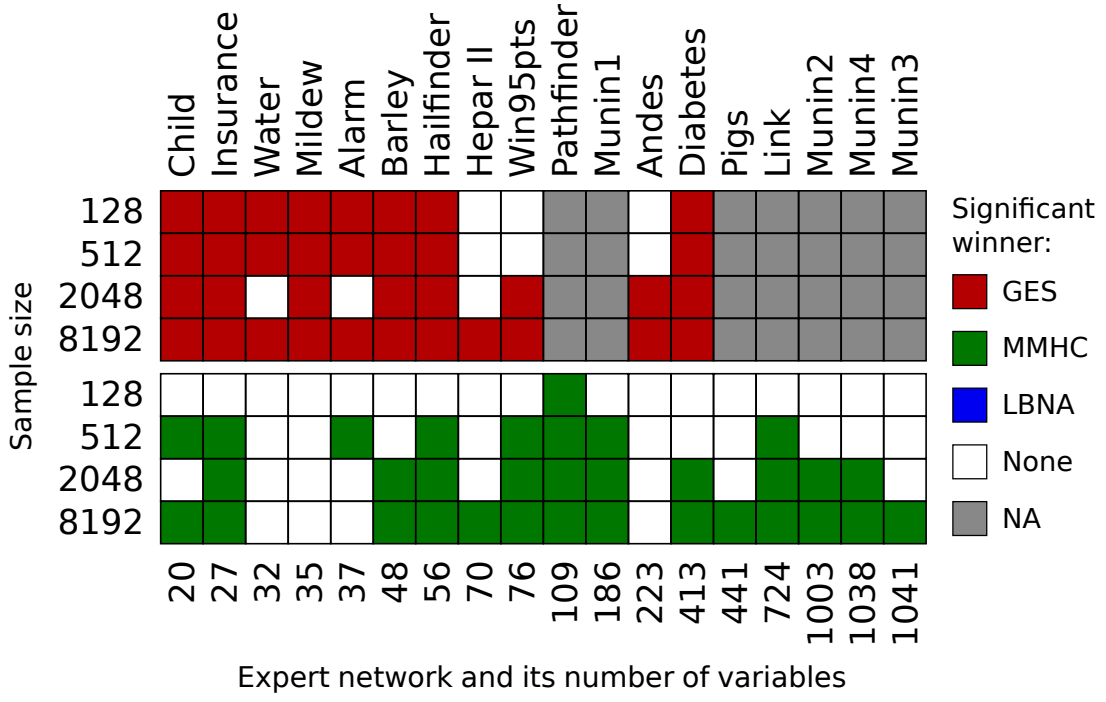


Figure 7.2.1.8: Prediction accuracy

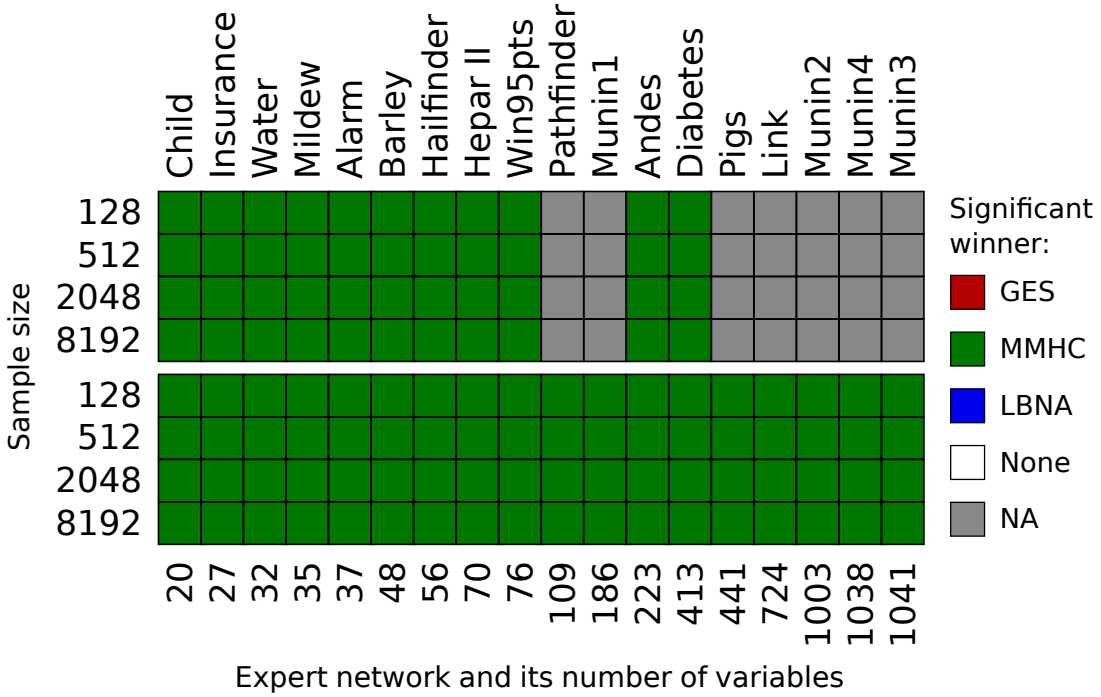


Figure 7.2.1.9: Execution time

7.2.2. Significant Winners—Analysis

As we can see from Figure 7.2.1.1 in the case of comparison of the SHD-based quality between all three algorithms the most often significant winner is the GES algorithm. The second is the LBNA method. However we do not know how this comparison would look for the case of biggest networks, where we did not obtain any results with the GES method. But in the case of comparison between only the MMHC and LBNA algorithms, where we have the complete table of statistics, we can derive a strong conclusion basing on this figure: the LBNA approach is significantly better than the MMHC method in almost all cases where any significant winner can be found. This dominance is especially visible in the second half of expert networks—these of biggest size.

Figures 7.2.1.3 and 7.2.1.5 shows clearly where lies the main strength of the LBNA approach. It is an outstanding from the two remaining methods, especially from MMHC, precision of the returned networks—both in the sense of the skeleton and v-structures detection. This seems to be a very natural result, if we remind ourselves what is the motivation and construction of the LBNA approach. It clearly aims at finding a sparse result, possibly sparser than it should be—but very precise, as even a one local network without a connection between some two vertices results according to the LBNA mechanism in not connecting these nodes in the global model.

This sparsity of the results returned by the LBNA approach comparing to the MMHC and GES methods is clearly confirmed in Figure 7.2.1.2, from which we can conclude that GES is producing significantly denser models than the two remaining methods, while in the comparison between MMHC and LBNA clearly MMHC returns denser networks.

From Figures 7.2.1.4 and 7.2.1.6 we can see that LBNA is the weakest approach in terms of the recall of the results—both in the sense of skeleton and v-structures detection. Namely

it is weaker in comparison with the MMHC approach, while the GES approach absolutely dominates both remaining methods. According to Figure 7.2.1.2 the GES approach finds significantly denser models than the two remaining algorithms, which is the opposite of the LBNA approach, as we have discussed it previously. What is important is that although the LBNA method in the case of almost all expert networks and sample sizes returns very sparse models comparing even to the MMHC approach, its precision gain is sufficiently high comparing to the recall loss in order to win in the sense of the SHD measure with the MMHC algorithm.

Results concerning the normalized score measure given in Figure 7.2.1.7 are rather not surprising. The global undisputed winner is here the GES method—a strong score-based approach, clearly aiming at maximizing this measure. On the second place is here the MMHC approach, while the LBNA method is weakest. The consistent criterion at least asymptotically with the growing size of the sample favors results which are able to model the generative distribution. Meanwhile the LBNA approach finds the sparsest results among the three compared methods, and as we can see according to the recall results sparser than they should be—thus its poor performance according to the normalized score could be in fact expected.

From the results presented in Figure 7.2.1.8 we can see that the strength in the prediction accuracy is correlated here with the results in the previous figure—however here there is quite often no any significant winner. The GES approach is the only one which is able to produce models of significantly better prediction accuracy than the two remaining methods. And again the MMHC approach is better than the LBNA method—which is here never a significant winner. Similarly as previously we can say: the sparsest models produced by the LBNA algorithm, although potentially closest to reality—especially in comparison to the MMHC approach, are not able to fit data so well as the denser networks produced by the two remaining methods.

7.2.3. Time Scalability

Figure 7.2.1.9 shows us that the MMHC method is overall dominating in the sense of the execution time. However, in order to better visualize how each of our three compared algorithms is time scalable with respect to the increasing size of the learned models we have additionally created pairs of plots, which can be seen in Figures 7.2.3.1, 7.2.3.2, 7.2.3.3 and 7.2.3.4. The left plots present for each of the considered expert network (they are ordered as everywhere else according to the number of contained variables) what was the average execution time of each of the three algorithms in the experiment corresponding to the network and for the sample size respectively 128, 512, 2048 and 8192. The biggest sample seems to be here the most pessimistic for the MMHC and LBNA approaches scenario—as in the case of smaller samples there are more often not performed statistical tests for conditional independence—the conditional independence is automatically assumed whenever the size of the sample occurs to be insufficiently large. Even in such scenario we can see that the GES approach is very poorly time scalable comparing to the two remaining methods. A similar situation can be seen in all remaining plots. Although there are no any results here (as everywhere else) about the GES behavior in the case of especially few biggest networks, this behavior might be rather clearly expected. This is a major drawback of the GES approach, resulting in its doubtful usefulness, regardless of the quality of the returned by this approach results—as we might deal in practice with even bigger number of variables than in all presented here experiments.

Although according to Figure 7.2.1.9 for each of the considered expert networks the MMHC approach is significantly faster than the LBNA approach—this significance is rapidly decaying with the increase of the learned models.

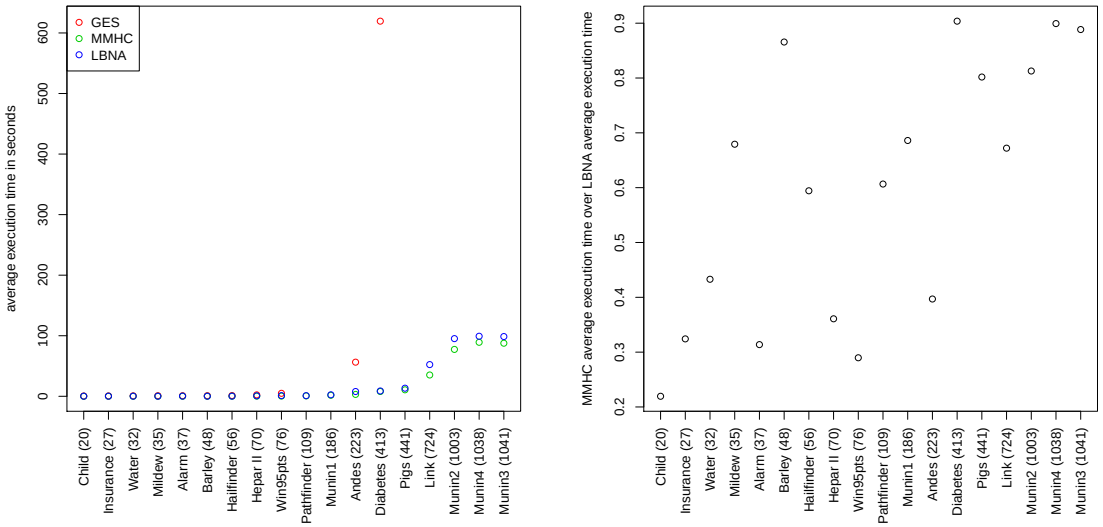


Figure 7.2.3.1: Execution time statistics for the case of sample size 128.

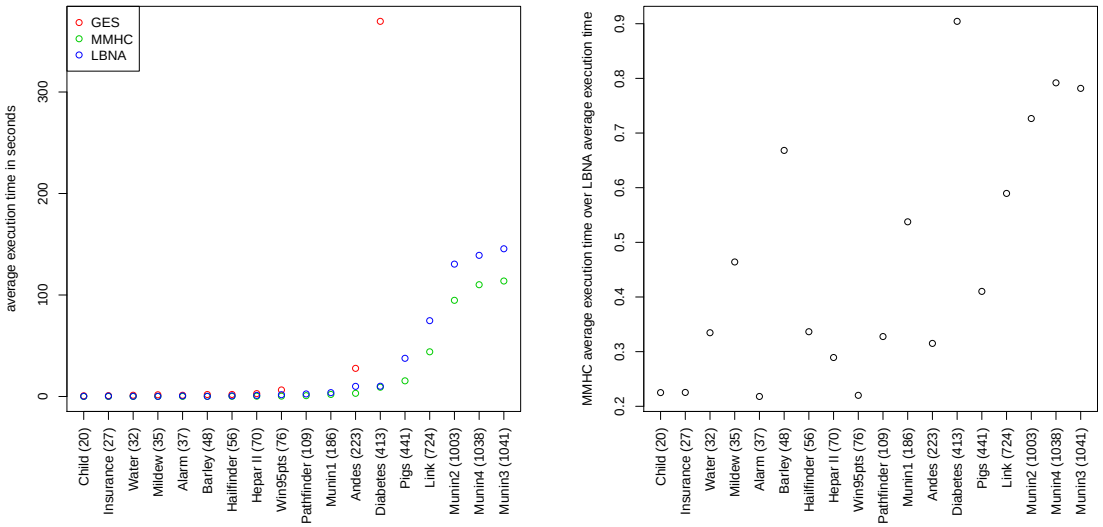


Figure 7.2.3.2: Execution time statistics for the case of sample size 512.

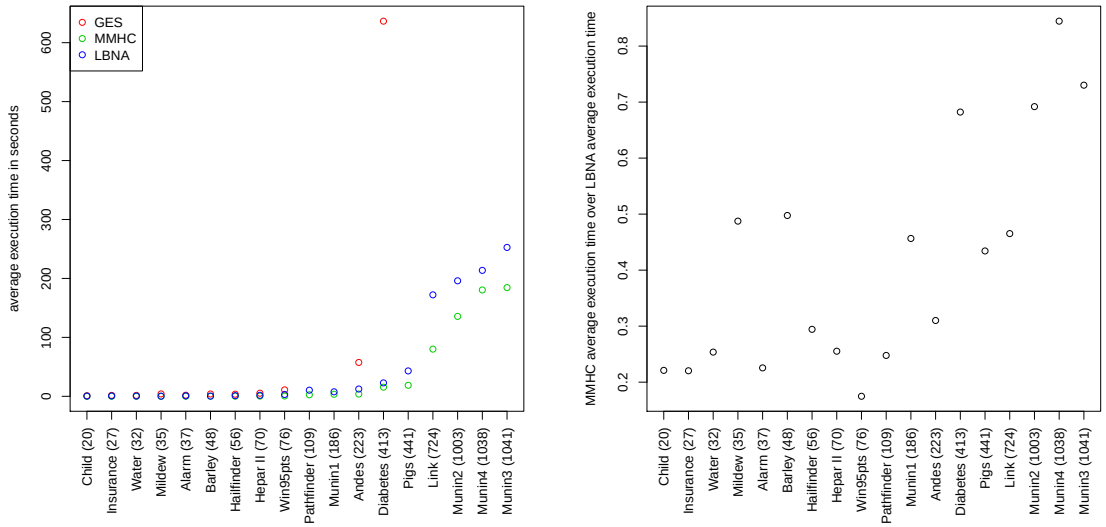


Figure 7.2.3.3: Execution time statistics for the case of sample size 2048.

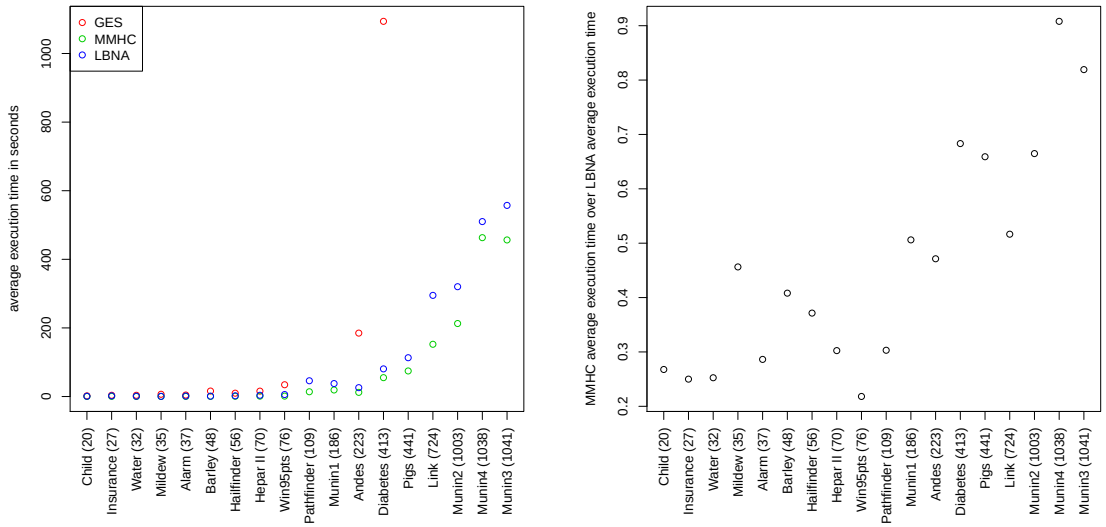


Figure 7.2.3.4: Execution time statistics for the case of sample size 8192.

We can see it exactly in the right plots of Figures 7.2.3.1, 7.2.3.2, 7.2.3.3 and 7.2.3.4. They present for each expert network what is the average execution time of the MMHC method divided by the average execution time of the LBNA method, where both averages are calculated like in the left plots—that is over the samples corresponding to the expert network and of size respectively 128, 512, 2048 and 8192.

In the case of the biggest considered here expert networks the difference between the MMHC and LBNA methods becomes in fact negligible—as the time ratio is here somewhere at the level between 0.8 and 0.9. Moreover, when we look at the shape of this plot, it seems to be possible that in the case of bigger expert networks the ratio would reach level close to 1. Of course it cannot exceed it, as the LBNA approach, that is LBNA(MMHC, MMHC), runs in fact the MMHC algorithm in order to find the initial global model from which the Markov blankets approximations are retrieved.

The time scalability results presented here give a strong support for our claims given in Subsection 6.5.6 about the time scalability with regard to the number of attributes of LBNA comparing to MMHC in the practical scenario SP_2—that is exactly in the scenario appearing here.

7.2.4. Small Sample Issues

Let us add one more important comment. It seems that more valuable results are these for the case of bigger sample sizes. In the case of these smallest there can be observed in the tables in Appendix A some strange at the first glance behavior of the MMHC and LBNA methods. Namely, they often achieve there better results than in the case of biggest sample sizes in the sense of the SHD metric.

The explanation lies in the nature of performed statistical tests for conditional independence. They automatically return a conditional independence whenever the sample size is insignificantly big—as we have pointed out it in Subsection 7.1.1, and also mentioned about it on the previous pages in this section. This can result in the case of these smallest sample sizes in the situation where the algorithms MMHC and LBNA return almost or even completely empty graphs, leading in particular to the unnaturally high value of the SHD metric—simply because the real model is very sparse too and regardless of the precision and recall of the resultant model there are not many edge modifications required leading from the completed PDAG of the learned structure to the original completed PDAG.

This phenomenon is clearly visible in the statistics in the appendix, namely for the normalized number of edges measure. For almost all expert networks this measure is increasing for MMHC and LBNA with the increase of the sample size. In the case of GES this behavior is less regular, however it is often exactly opposite.

Some more valuable results in the case of smallest samples could be potentially obtained with an assist of the Fisher's exact test for conditional independence (mentioned in Subsection 2.2.4), where the calculation of this test would be still feasible, and in the same time we would receive on the basis of the available sample much more reliable knowledge about the conditional independencies. Unfortunately, this test is not available in the *bnlearn* package.

Another odd behavior in the case of the small sample size is related also to the GES method. Its execution time in the case of networks Andes and Diabetes looks strange, as the GES method is here slow for the sample size 128—comparing to the bigger sample 512. The following explanation might be correct. The GES method in the case of smaller sample sizes clearly overfits the available data, as it can be in fact noticed in the tables corresponding to the normalized score for many considered here expert networks. The normalized score is often in such cases much less than 1, indicating that the GES algorithm learns graphs of

higher score than the true expert network achieves. This behavior is sometimes supported with the additional normalized number of edges factor—which, as we have mentioned it earlier, often is decreasing for GES with the increase of the sample size. In other words this overfitting in the case of smallest sample sizes can additionally result in returning much denser models than required. Andes and Diabetes are the two biggest networks on which we were still able to perform the calculation with the GES method in the limit of 6 GB memory. So possibly the GES overfitting here for the unreliable sample size 128 leads to some problematic searching in the space of quite big here models. Especially in the case of the Andes network this explanation is confirmed in statistics, as the behavior of both the normalized score and normalized number of edges measure is clearly consistent here with the overall trend.

What is important is that all the claims appearing in this section, in particular in Subsection 7.2.2 and 7.2.3, hold clearly in the case of these most reliable, that is biggest considered here sample sizes.

7.3. Implementation of LBNA

In this section we describe in details how all the performed and described in the previous sections experiments can be repeated. We list all necessary prerequisites, and we describe the most fundamental functionality of the created *R* code corresponding to the implementation of LBNA and described in the previous sections experiments comparing GES, MMHC and LBNA.

Note that the repetition of all experiments does not mean replicating them. The experiment scheme applied here and defined in Section 7.1.7 have clearly randomized nature. Each of the training and test dataset is generated from the expert network randomly—and so as a consequence any repetition of the experiments leads to the generation of completely new datasets. However, for each considered setting (the expert Bayesian network, and sample size) we generate 128 samples, and moreover for each considered quality measure we perform the statistical comparison of the evaluated on these tables algorithms with an assist of the permutation test. Consequently it should be expected that in particular most of the statistically significant conclusions visualized in Figures from 7.2.1.1 to 7.2.1.9 should remain unchanged, regardless of the independent repetition of all the experiments.

7.3.1. Prerequisites

Let us first start with the prerequisites. Most of the functionalities enabling performing the experiments are written in the *R* language and invoked with the assist of *R* environment (R Development Core Team, 2014). Namely, we have used *R* version *3.1.2*. Three *R* packages must be additionally installed. Below we list them, together with the versions used in our experiments:

- *bnlearn 3.8.1*—this package contains a rich functionality for learning and manipulating Bayesian networks, visualizing them and performing inference with them. In particular it contains the implementation of the MMHC algorithm used in the experiments.

- *coin 1.0-24*—we use the implementation of the two-sided permutation test included in this package, in order to perform the statistical comparison between the compared algorithms and in particular detect significant winners according to any of the nine considered measures.

- *plyr 1.8.2*—we use some functions manipulating R data structures included in this package.

The Java Runtime Environment is also required. We have used *OpenJDK Runtime Environment 2.6.6*.

Some other versions of the *R* environment, as well as the packages and Java Runtime Environment can be possibly used. Note however, that we cannot assure the compatibility of the used here implementations of the tested algorithms for such different setting. We recommend using exactly the above listed configuration.

After installing the appropriate versions of the *R* software, mentioned packages and Java Runtime Environment create one folder and copy there all the files available in the folder `implementation` submitted together with the dissertation. Alternatively request the author for this folder via the `pbetl@mimuw.edu.pl` email. It contains the following files:

- 18 `rda` files, containing stored in the appropriate format used in the *bnlearn* package expert Bayesian networks applied in the experiments and defined in Subsection 7.1.3, that is respectively: `child.rda`, `insurance.rda`, ..., `munin3.rda`. All these files are also available on the web page `http://www.bnlearn.com/bnrepository/`.
- `tetradcmd-5.1.0-10.jar`—the command line version of the TETRAD software, containing in particular the implementation of the GES algorithm used in the experiments. It can be also downloaded from the web page `http://www.phil.cmu.edu/projects/tetrad_download/download/`.
- `lbna.r`—our implementation of the LBNA practical version (Algorithm 6.4.3.1), of the whole experiment scheme defined in Subsection 7.1.7 and of some of the auxiliary functionalities is included in this *R* source file.

7.3.2. Loading Networks and Generating Datasets

After launching *R* in the working directory corresponding to the folder where all the above listed files have been placed, type:

```
> source("lbna.r")
```

in order to read the content of this file. Let us first explain the most basic functions included there.

The function `read.network` defined in the `lbna.r` file as:

```
read.network = function(file)
```

reads the content of the `rda` file the path to which is given in the `file` argument and returns the list of two elements:

- `network`—the object of type `bn.fit` representing exactly the Bayesian network encoded in the indicated file. Class `bn.fit` is the representation of Bayesian networks in the *bnlearn* package.
- `graph`—the object of type `bn` representing the Bayesian network structure (that is the DAG without parameters)—corresponding to element `network`. Class `bn` is the representation of Bayesian network structures in the *bnlearn* package. The names of the nodes in the order kept in the object (corresponding to the ancestral order of nodes) are converted in `graph` from the original character strings included in `network` to character strings representing consecutive positive integers, that is `"1"`, `"2"`, `"3"` and so on.

The function `generate.table` defined in the `lbna.r` file as:

```
generate.table = function(network, m)
```

generates the object of type `data.frame` representing the sample of size given in the parameter `m` generated from the Bayesian network (some object of class `bn.fit`) given in the parameter `network`. The names of the consecutive columns of this table, corresponding to the ancestral order of nodes kept in `network`, are converted to character strings representing consecutive positive integers, that is `"1"`, `"2"`, `"3"` and so on. Each column is the object of type `factor`, having some finite number of possible values called in R levels. The levels of the returned table are converted to consecutive nonnegative integers: `0`, `1`, `2` and so on.

As a consequence, if we for example type the following instructions:

```
> x = read.network("alarm.rda")
> g = x$graph
> d = generate.table(x$network, 8192)
```

then the dataset `d` is a sample of size 8192 (the largest considered in the described experiments size) generated from the distribution represented by the Bayesian network called Alarm, which structure is kept in the object `g`. The table `d` can be used then to learn the network structure with the assist of some Bayesian network structure learning algorithm. The obtained result can be compared with the perfect solution held in object `g`.

7.3.3. Learning Networks from Datasets with GES, MMHC and LBNA

Many Bayesian network structure learning algorithms are implemented in the *bnlearn* package, in particular MMHC. For the purpose of experiments we have created three functions corresponding exactly to the three investigated in this chapter methods in versions specified in Subsections 7.1.1 and 7.1.2: GES, MMHC and LBNA(MMHC, MMHC). They are respectively defined in the `lbna.r` file as follows:

```
ges.bdeu = function(data)
mmhc.bdeu.g2adf = function(data)
lbna.mmhc.mmhc = function(data)
```

The parameter `data`—the object of type `data.frame`—contains some dataset for which the Bayesian network structure, that is the object of type `bn`, is learned and returned.

The function `ges.bdeu` invokes the GES method included in `tetradcmd-5.1.0-10.jar` with the default scoring criterion, that is the BDeu score with the equivalent sample size 10. The function `mmmhc.bdeu.g2adf` invokes the implemented in the *bnlearn* package MMHC algorithm with the appropriately set parameters—that is with the BDeu score with the equivalent sample size 10 used in the Hill-Climbing part and the G^2 test with adjusted degrees of freedom df^T_{adj} applied in the MMPC subroutine.

The function `lbna.mmhc.mmhc` is our implementation of Algorithm 6.4.3.1, with the auxiliary methods $\mathcal{MC}$ and $\mathcal{MB}$ set to MMHC—that is to the `mmhc.bdeu.g2adf` function.

On the basis of our exemplary dataset `d` Bayesian network structure with an assist of any of the above algorithms, that is an object `h` of type `bn` , can be obtained with an assist of one of the following commands:

```
> h = ges.bdeu(d)
```

or

```
> h = mmhc.bdeu.g2adf(d)
```

or

```
> h = lbna.mmhc.mmhc(d)
```

7.3.4. Evaluating Learned Networks

The learned models can be evaluated with regard to the nine criteria defined in Subsections 7.1.4, 7.1.5 and 7.1.6 in the following way:

- We can calculate the Structural Hamming Distance `str.ham.dist` between the real structure `g` and the learned structure `h` with an assist of the `shd` function implemented in the *bnlearn* package (see http://www.bnlearn.com/documentation/man/compare.html), in the following way:

  ```
  > str.ham.dist = shd(g, h)
  ```

- We can calculate the skeleton precision `skel.prec` and recall `skel.rec` measures of the learned structure `h` with regard to the real structure `g` with an assist of the `skeleton.similarity` function, which is defined in the `lbna.r` file as:

  ```
  skeleton.similarity = function(learned.graph, original.graph)
  ```

  The `skeleton.similarity` function takes as the parameters two DAGs operating on the same set of vertices, and returns the list of two elements:

  - `precision`—the skeleton precision measure of the `learned.graph` model with regard to the graph `original.graph`,
  - `recall`—the skeleton recall measure of the `learned.graph` model with regard to to the graph `original.graph`.

  Thus, in order to obtain the `skel.prec` and `skel.rec` values the following commands can be invoked:

  ```
  > skel.sim = skeleton.similarity(h, g)
  > skel.prec = skel.sim$precision
  > skel.rec = skel.sim$recall
  ```

- We can calculate the v-structures precision `vstructs.prec` and recall `vstructs.rec` measures of the learned structure `h` with regard to the real structure `g` with an assist of the `vstructures.similarity` function defined in the `lbna.r` file, which has the analogical input and output as the `skeleton.similarity` function, with the only difference that here in the returned list the elements `precision` and `recall` correspond respectively to the v-structures precision and recall measure. Thus, the following commands are enough to calculate the `vstructs.prec` and `vstructs.rec` values:

  ```
  > vstructs.sim = vstructures.similarity(h, g)
  > vstructs.prec = vstructs.sim$precision
  > vstructs.rec = vstructs.sim$recall
  ```

- We can derive the normalized number of edges measure `norm.numb.of.edges` of the learned structure `h` with regard to the real structure `g` with an assist of the function `arcs` included in the *bnlearn* package (see http://www.bnlearn.com/documentation/man/mb.html), which for the given Bayesian network structure (an object of class `bn`) returns the matrix, where each row indicates one of the edges appearing in the structure. We can derive the number of edges in the structure by counting the number of rows of such matrix, which can be done with the assist of the function `nrow` implemented in the package *base* (included always in *R*, see https://stat.ethz.ch/R-manual/R-devel/library/base/html/nrow.html). As the result the desired measure is achievable as follows:

```
> norm.numb.of.edges = nrow(arcs(h)) / nrow(arcs(g))
```

- The normalized score measure `norm.score` of `h` with regard to `g` and the training dataset `d` can be easily derived with an assist of the function `score` implemented in the *bnlearn* package (see http://www.bnlearn.com/documentation/man/score.html). This function enables us to calculate the BDeu score with the equivalent sample size 10 on the dataset `d` of graphs `g` and `h`—respectively `score.d.g` and `score.d.h`—in the following way:

```
> score.d.g = score(g, d, type = "bde")
> score.d.h = score(h, d, type = "bde")
```

  Then it is easy to calculate the normalized score:

```
> norm.score = score.d.h / score.d.g
```

- In order to obtain the prediction accuracy measure `pred.acc` evaluating the learned graph `h` we first of all have to generate from the distribution encoded in the `x$network` network the additional testing dataset, let us call it `td`. In the described experiments we generate the test sample of size 1024—let us use the same size here:

```
> td = generate.table(x$network, 1024)
```

  Then we have to convert the learned structure `h` into a Bayesian network by adjusting the parameters encoding the conditional distributions of each vertex in `h` given its parents on the basis of the observed frequencies in the training dataset `d`. This can be achieved with the `bn.fit` function implemented in the *bnlearn* package (see http://www.bnlearn.com/documentation/man/bn.fit.html). Namely, in order to convert `h` into the corresponding to it Bayesian network `bn.h` (the object of type `bn.fit`), we execute the following command:

```
> bn.h = bn.fit(h, d)
```

  The prediction accuracy measure of `bn.h` can be directly obtained with an assist of the `prediction.accuracy` function defined in the `lbna.r` file as:

```
prediction.accuracy = function(network, data)
```

which for the parameter `network` of type `bn.fit`, that is for the tested Bayesian network, and for the parameter `data` of type `data.frame` corresponding to the testing dataset, returns the desired measure value. Thus, in order to obtain `pred.acc` value we run the following command:

```
> pred.acc = prediction.accuracy(bn.h, td)
```

- In oder to calculate the execution time measure `exec.time` we can invoke the function `system.time` for the process of learning the network. The `system.time` function implemented in the package *base* (see https://stat.ethz.ch/R-manual/R-devel/library/base/html/system.time.html) returns the vector of several values characterizing the investigated process, including consecutively the CPU time, system time, and elapsed time. We stick to this third value, as the CPU time does not take into account the CPU time of the invoked process outside of *R*—which is exactly the situation in the case of the `ges.bdeu` function, where the external TETRAD software is used. Thus, in order to obtain the execution time measure of each of the three considered Bayesian network structure learning algorithms, we respectively run:

  ```
  > exec.time = system.time(h <- ges.bdeu(d))[3]
  ```

  or

  ```
  > exec.time = system.time(h <- mmmhc.bdeu.g2adf(d))[3]
  ```

  or

  ```
  > exec.time = system.time(h <- lbna.mmhc.mmhc(d))[3]
  ```

7.3.5. Experiments Automation

Fortunately, in order to repeat all the described in this chapter experiments a potential user does not have to proceed through all the commands given in Subsection 7.3.4 for each of the 72 experiments, corresponding to 18 used expert Bayesian networks and 4 considered sample sizes, and within each experiment for 128 randomly generated training and test datasets for each of the three compared algorithms. We have partially simplified this process by means of defining in the `lbna.r` file two higher level functions enabling performing much more automatically all the experiments, each of them strictly corresponding to the whole procedure defined in Subsection 7.1.7.

The first function is called `experiments.set.given.network.and.method`. It is defined in the `lbna.r` file as:

```
experiments.set.given.network.and.method = function(output.file,
        network.file, train.sample.size, test.sample.size, repetitions,
        learning.method)
```

This function performs for the expert Bayesian network encoded in the `rda` file the path to which is given in the parameter `network.file` and for each sample size of generated datasets occurring in the numeric vector `train.sample.size` Step 1 of the experiment scheme defined in Subsection 7.1.7—for only one specified in the parameter `learning.method` Bayesian network structure learning algorithm in the role of $\mathcal{M}$ (it can be set to `ges.bdeu` or `mmhc.bdeu.g2adf` or `lbna.mmhc.mmhc`).

The parameter `test.sample.size` is the numeric vector of the same length as the vector `train.sample.size`. Its i-th element specifies the size of the generated test samples (used in the calculations of the prediction accuracy measure) in the experiment corresponding to training samples of size given in i-th element of the `train.sample.size` vector. In our experiments we have always used the value 1024 of the test samples in each performed experiment, regardless of the size of training samples.

The parameter `repetitions` specifies the number of generated training datasets used to evaluate the performance of the method `learning.method`. In the case of our experiments this parameter has been always set to 128.

All gathered statistics during each calculation of Step 1 of the experiment scheme for each specified in parameters setting are saved to the file the path to which is specified in the parameter `output.file`. We do not go into any details of the returned structures—they can be found in the comments in the `lbna.r` file. Let us only mention here that these output structures will be the entry of the second higher level function which aim is to perform all the statistical comparisons between considered learning methods (that is Step 2 of the experiment scheme defined in Subsection 7.1.7 for each considered setting).

For example, in order to gather all the statistics given in the fifth table appearing in Appendix A except the statistical comparison information corresponding to all considered in this chapter experiments performed on datasets generated from the Alarm network, we need to perform only three commands:

```
> experiments.set.given.network.and.method("alarm_ges", "alarm.rda",
c(128, 512, 2048, 8192), c(1024, 1024, 1024, 1024), 128, ges.bdeu)
> experiments.set.given.network.and.method("alarm_mmhc", "alarm.rda",
c(128, 512, 2048, 8192), c(1024, 1024, 1024, 1024), 128, mmhc.bdeu.g2adf)
> experiments.set.given.network.and.method("alarm_lbna", "alarm.rda",
c(128, 512, 2048, 8192), c(1024, 1024, 1024, 1024), 128, lbna.mmhc.mmhc)
```

As the result in the files `alarm_ges`, `alarm_mmhc` and `alarm_lbna` we gather all statistics corresponding to achieved by respectively GES, MMHC and LBNA measures of quality on each generated from the Alarm network sample for each considered size of these samples.

7.3.6. Statistical Comparison and Results Export

In order to calculate the statistical comparison between methods and determine the potential significant winners basing on the statistics gathered in the three above mentioned files it is sufficient to invoke one function, called `produce.result.table`. This function is defined in the `lbna.r` file as:

```
produce.result.table = function(output.file, mmhc.file, lbna.file,
ges.file = NULL)
```

The parameters `mmhc.file`, `lbna.file` and `ges.file` should be the paths to the files containing the resultant statistics of respectively MMHC, LBNA and GES generated in the series of experiments conducted with the `experiments.set.given.network.and.method` function with the common parameters settings (except the `learning.method`—set for each of the three methods).

The parameter `ges.file` is an optional one. If it is not given the method assumes that we want to compare only MMHC and LBNA algorithms in the context of considered experiments. Recall that in the case of some experiments described in previous sections there was insufficient memory to perform learning with GES. As the result in some of the tables corresponding

to especially these biggest expert Bayesian networks in Appendix A there are compared only MMHC and LBNA algorithms. In such situation the function `produce.result.table` was invoked without setting the `ges.file` parameter.

The nice property of the `produce.result.table` function is that it writes to the file the path to which is specified in the parameter `output.file` the Latex source of the one page of tables exactly in the format occurring in Appendix A—describing the whole chunk of experiments corresponding to one considered here expert network.

For example, in order to generate and write to the file `alarm_results.txt` the Latex source of the page in Appendix A containing the tables of results for the Alarm network, assuming that the previously mentioned files `alarm_ges`, `alarm_mmhc` and `alarm_lbna` have been already created, it is sufficient to run the command:

```
> produce.result.table("alarm_results.txt", "alarm_mmhc", "alarm_lbna",
"alarm_ges")
```

The content of the produced file `alarm_results.txt` can be then placed in some Latex document. Note that one specific Latex package called *tabularx* is used in this generated code—so it must be included in the created document.

Chapter 8

Conclusions and Future Research

There are two main contributions of this work. The first one is the proposition of the significantly new view on the NP-hardness of the Bayesian network structure learning problem. Namely, we have proved that the problem of finding the network structure containing the least number of edges and representing the joint probability distribution of the sample is NP-hard. The formulation of the problem is simple comparing to the classical understanding of the Bayesian network structure learning hardness, requiring some more advanced background in the statistics domain. Moreover, it corresponds to a strictly practical scenario.

However, as we have mentioned it at the end of Chapter 3, the achieved result is not satisfactory for us, as several its enhancements would be very welcome. From the one side one can think about changing the optimization criterion to some more reasonable, like the minimal dimension of the learned structure, and from the other side some approximation versions of the current statement would be very interesting to consider. We have listed such possible versions in Subsection 3.3.5. Possibly we will focus in the future on investigating the hardness of several such modifications of the basic obtained here result.

The second contribution is a theoretical result showing how the global, that is based on all attributes, faithful representation of some joint probability distribution can be inferred from the appropriate—based on all Markov blankets—local faithful models. There has been also proposed an algorithm basing on this result, called LBNA (Local Bayesian Networks Aggregation). Its aim is to learn a global Bayesian network with an assist of the previously determined local networks basing on the subsets of attributes corresponding to all Markov blankets—where finding the approximations of these blankets is the first step of the whole process.

The proposed approach is a significant generalization of the methods designed by Margaritis and Thrun (2000) and Pellet and Elisseeff (2008). In these two works there has been proposed a mechanism for inferring a global faithful model from the appropriately acquired knowledge about the conditional independencies occurring inside previously found Markov blankets of all variables. In our work instead of the fixed mechanism of gathering this local knowledge we give a very big flexibility in this step. A user can choose not only any method of Markov blankets detection used in the initial step, but also any Bayesian network structure learning algorithm in the step of gathering local knowledge from these blankets. Such flexibility can potentially lead to new and interesting solutions.

We have tested and evaluated in many aspects one such possible setting, namely the LBNA(MMHC, MMHC) configuration, where the MMHC algorithm is first used in order to find a global Bayesian network from which initial Markov blankets approximations are derived, and then the MMHC method is again used in order to find all local Bayesian networks

corresponding to these blankets. The most significant comparison of this setting is with the MMHC approach. It clearly proves what is the value of the LBNA methodology, showing that when we adopt it to the MMHC approach we are able to reach something more than the MMHC itself.

As we have seen in the experiments section the LBNA(MMHC, MMHC) method is similarly well time scalable as the MMHC approach in the case of learning larger networks, and it returns in the same time most often significantly closer to the reality models in the sense of the SHD metric. Its main strength is high precision of the results. The main disadvantage is its low recall, resulting in the better usefulness of the MMHC approach if we are interested in finding models well fitting data, thus models of better classification abilities and higher scored. Even a better approach in these aspects has turned out to be the third compared here method, that is the GES algorithm. But this approach is much less useful in practice than the first two methods because of its very poor time scalability with respect to the increasing size of learned models.

There are numerous aspects of possible further research in the presented area. We finish with showing few of them.

First of all, as we have pointed out it and gave the appropriate example just after the proof of Proposition 6.2.1.1, in Subsection 6.2.2, there are possible several modifications of the proposition itself, that is conversions of the presented recipe for induction of both the skeleton and v-structures of a global model into a slightly different—but fully equivalent one. The equivalence is however only in the theoretical conditions guaranteed by the proposition, and it does not mean equivalence in the real scenario of applying the recipe in order to learn a global network from some existing or generated dataset. Let us for example recall optional Step 3 in the theoretical version of LBNA—that is in Algorithm 6.3.1.1. This step is completely optional in theory—but in practice its execution may lead to another behavior of the method. LBNA is based on one of at least few such possible versions of the theoretical recipe, and presumably using other versions could result in the method of significantly different empirical properties.

We would be especially interested in a modification which could lead us to the LBNA methodology improved most of all in the recall of returned results, as this is undoubtedly according to the performed experiments the weakest spot of our approach.

Besides the change of the theoretical recipe on the basis of which the LBNA algorithm is implemented there are some other modifications possible. Notice that we have applied a rather simple greedy algorithm for the case of merging information about v-structures from local models. Also the final learning of some concrete global Bayesian network in the LBNA procedure is based on the simple Hill-Climbing approach, similar to the one used in the MMHC method, although with more initial restrictions resulting from the a priori determined set of v-structures. Replacing all these greedy approaches with some more advanced procedures could potentially also give some increase in the quality of obtained results—although we must keep in mind that the efficiency of the proposed solution is also a very important factor.

Last but not least, several more settings of the LBNA approach could be tested. As we have already stated it previously, the LBNA method offers a big flexibility. The experiments presented in this dissertation indicate that it is worth to search for such other settings—as the potential of obtaining some new interesting results is high. For example one can considered applying some strong but slow method in the role of the $\mathcal{MC}$ auxiliary tool, that is the method used to learn local networks in the LBNA procedure. It seems that one of the interesting possibilities here could be the GES algorithm. In the case when the local networks are small the poor time performance of the method might be less important, while the obtained local results might be significantly better. We must be here careful however. Let us recall one of the used in experiments expert Bayesian network, called Pathfinder. One of

the Markov blankets corresponding to this structure contains almost all attributes. In such case learning the local structure is in fact the same problem as learning the whole structure. In other words one Markov blanket outlier may cause a huge efficiency loss if we apply such poorly scalable method in the role of local structures learning tool like GES. In order to bypass somehow this problem we might think about some variable strategy depending on the size of the blanket, applying the more advanced and slower local structures learning approach only to the blankets of some reasonably small size.

There is also an another interesting possibility. Namely, we can think about employing domain experts in order to resolve the local structures. LBNA splits the problem of learning a whole potentially very big network into determining typically much smaller local models—and this is exactly the place where the domain experts could be very helpful. It is indeed a much easier task for them to build such local model than the global one, where in order to detect the set of direct reasons (parents) of any vertex it is sufficient to scan some subset of remaining attributes instead of all of them.

Let us end with a more general comment. The proposed here methodology, similarly as all the remaining strongly related to it constraint-based solutions, is relaying on one basic assumption—the faithfulness of the generative distribution. Of course in practice we can use these methods regardless of the satisfaction of this assumption, which is in fact even not verifiable. But the theoretical correctness of these solutions is essentially based on this assumption.

In particular without this assumption there is no guarantee that there exists a unique Markov blanket for a given variable—while in LBNA as well as in several other discussed related approaches the splitting of the problem is with regard to one Markov blanket of each variable, or more generally one local subset of attributes around each variable, like the set of neighbors in the faithful representation in the case of MMPC. These algorithms exploit very effectively the simplifying assumption of faithfulness and its consequences, but they are not prepared to face such more general scenario. One of the possible directions for the future is thus to try to redesign the whole LBNA methodology in order to deal somehow with this case.

Appendix A

Detailed Results of Experiments

Each of the following 18 tables corresponds to one of the 18 performed set of experiments, for all the possible 18 choices of the expert Bayesian network used in the role of the samples generator. Subsection 7.1.3 gives some overview of all these applied networks. For each expert network the corresponding table (the one with the caption containing the name of the network) presents all the statistics gathered during the conducted experiment for each of the 4 applied sample sizes: 128, 512, 2048 and 8192. For the given expert network and sample size the experiment was conducted exactly as it is explained in the Subsection 7.1.7.

Each of the 18 tables consists of nine subtables corresponding to the nine calculated for the given expert network quality measures of compared algorithms: Structural Hamming Distance, normalized number of edges, skeleton precision, skeleton recall, v-structures precision, v-structures recall, normalized score, prediction accuracy and execution time. Subsections 7.1.4, 7.1.5 and 7.1.6 give a detailed description of each of these measures.

Each of the nine subtables has the same construction. In the most of the 18 reported here choices of a sample generator—expert network there were compared three algorithms: GES, MMHC and LBNA(MMHC, MMHC). All details about their implementations can be found in Subsections 7.1.1 and 7.1.2. We will further write LBNA instead of LBNA(MMHC, MMHC)—and this convention will be also used in all further tables. In the case of some networks however, in particular these biggest, there was insufficient available memory in order to perform the GES algorithm—even for the smallest sample size 128. The limit for each of the three compared algorithms was set to 6 GB of memory. Whenever the GES approach exceeded this limit the nine subtables are simplified in order to express the comparison only between the MMHC and LBNA algorithms.

In the case of the expert networks where all three algorithms are compared, each of the nine subtables consists of columns expressing the following information about the corresponding to it quality measure:

- Sample—size of the sample.
- GES—for each of the considered sample sizes this column expresses firstly the average and then in the bracket the standard deviation of the measure in the case of 128 repetitions of GES calculation on the random samples of the given size generated from the expert network.
- MMHC—the same as the above, but calculated for the MMHC approach.
- LBNA—the same as the above, but calculated for the LBNA approach.

- GES-MMHC—for each of the considered sample sizes it expresses the p-value of the permutation test comparing the 128 measure values calculated over the random samples of the given size for the GES and MMHC algorithms.
- GES-LBNA—the same as the above, but here the p-value expresses the comparison between the GES and LBNA approaches.
- MMHC-LBNA—the same as the above, but here we compare the MMHC and LBNA approaches.
- Significant winner—for each of the considered sample sizes, whenever one of the three compared methods obtains according to the 2nd, 3rd and 4th column on average better results of the measure than the remaining two methods on the samples of the given size (better means higher in the case of: normalized number of edges, skeleton/v-structures precision/recall and prediction accuracy, while better means lower in the case of Structural Hamming Distance, normalized score and execution time), and the two p-values corresponding respectively to the comparison of this method with each of the two remaining are both less than 0.05, then such method is treated as a significant winner, and it appears for the considered sample size in this column. If there is no any significant winner we place instead the "-" sign.

  In the case of the normalized number of edges the notion of the significant winner is here in fact artificial. There is no any reason to claim without any additional knowledge that denser or sparser results are better. In the case of all remaining measures there is no such problem, as their interpretation with respect to the goodness of the returned model is clear. In order to adopt somehow this measure in the proposed scheme we have decided to treat a method returning significantly denser results as the significant winner.

  We also additionally check whether the LBNA method significantly wins with the MMHC (corresponding p-value in the column MMHC-LBNA less than 0.05 and LBNA better on average than MMHC), or MMHC significantly wins with the LBNA (the same p-value less than 0.05 and MMHC better on average than LBNA). If one of these two methods significantly wins over the second—we put it in the bracket at the same column and for considered sample size, just after placing the information about the global (calculated between all three algorithms) significant winner. If neither the MMHC nor the LBNA approach significantly wins we put in the bracket the "-" sign.

In the case of the expert networks where only the MMHC and LBNA methods are compared each of the nine subtables consists of the following columns: Sample, MMHC, LBNA, MMHC-LBNA, Significant winner. Their meaning is exactly the same as in the bigger tables corresponding to the comparison between all three methods. The only one slight difference appears in the case of the last column. Here the significant winner is calculated only between two methods: MMHC and LBNA. So the information appearing here corresponds exactly to the one given in the brackets in this column for the case of comparison between three methods.

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	21.414 (4.15)	26.93 (2.411)	26.648 (2.176)	0	0	0.34203053	GES (-)
512	8.938 (3.502)	31.523 (3.353)	27.234 (3.003)	0	0	0	GES (LBNA)
2048	0.938 (1.625)	12.203 (3.359)	6.867 (5)	0	0	0	GES (LBNA)
8192	0.102 (0.432)	27.32 (3.265)	15.383 (3.391)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.073 (0.083)	0.321 (0.075)	0.319 (0.068)	0	0	0.86096478	GES (-)
512	0.975 (0.047)	0.829 (0.085)	0.63 (0.074)	0	0	0	GES (MMHC)
2048	1.007 (0.024)	1.001 (0.044)	0.955 (0.04)	0.23081303	0	0	- (MMHC)
8192	1.002 (0.009)	1.213 (0.062)	0.857 (0.092)	0	0	0	MMHC (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.716 (0.066)	0.514 (0.182)	0.526 (0.174)	0	0	0.56880188	GES (-)
512	0.924 (0.044)	0.46 (0.099)	0.587 (0.126)	0	0	0	GES (LBNA)
2048	0.99 (0.019)	0.96 (0.048)	0.998 (0.009)	0	3.529e-05	0	LBNA (LBNA)
8192	0.998 (0.009)	0.592 (0.039)	0.839 (0.077)	0	0	0	GES (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.765 (0.059)	0.16 (0.056)	0.166 (0.059)	0	0	0.43323898	GES (-)
512	0.9 (0.037)	0.383 (0.098)	0.37 (0.09)	0	0	0.25406647	GES (-)
2048	0.997 (0.011)	0.96 (0.038)	0.953 (0.039)	0	0	0.21963406	GES (-)
8192	1 (0)	0.716 (0.029)	0.714 (0.046)	0	0	0.65071678	GES (-)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.19 (0.116)	0.734 (0.443)	0.742 (0.439)	0	0	1	- (-)
512	0.639 (0.178)	0.179 (0.301)	0.463 (0.443)	0	3.529e-05	0	GES (LBNA)
2048	0.939 (0.105)	0.611 (0.155)	0.813 (0.185)	0	0	0	GES (LBNA)
8192	0.992 (0.035)	0.291 (0.17)	0.764 (0.216)	0	0	0	GES (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.308 (0.164)	0.002 (0.018)	0.002 (0.018)	0	0	1	GES (-)
512	0.652 (0.171)	0.055 (0.096)	0.07 (0.099)	0	0	0.24940777	GES (-)
2048	0.978 (0.063)	0.711 (0.146)	0.844 (0.153)	0	0	0	GES (LBNA)
8192	1 (0)	0.283 (0.116)	0.592 (0.143)	0	0	0	GES (LBNA)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.986 (0.004)	1.218 (0.024)	1.219 (0.025)	0	0	0.66296577	GES (-)
512	0.998 (0.001)	1.199 (0.03)	1.223 (0.028)	0	0	0	GES (MMHC)
2048	1 (0)	1.02 (0.022)	1.026 (0.031)	0	0	0.11658955	GES (-)
8192	1 (0)	1.146 (0.013)	1.206 (0.035)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.781 (0.015)	0.658 (0.019)	0.657 (0.021)	0	0	0.5240202	GES (-)
512	0.812 (0.012)	0.718 (0.023)	0.701 (0.023)	0	0	0	GES (MMHC)
2048	0.819 (0.011)	0.81 (0.017)	0.808 (0.018)	1.91e-06	0	0.23128128	GES (-)
8192	0.818 (0.012)	0.761 (0.013)	0.733 (0.023)	0	0	0	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.506 (0.063)	0.036 (0.001)	0.164 (0.583)	0	0	0	MMHC (MMHC)
512	0.705 (0.074)	0.052 (0.003)	0.231 (0.023)	0	0	0	MMHC (MMHC)
2048	1.044 (0.064)	0.086 (0.01)	0.389 (0.026)	0	0	0	MMHC (MMHC)
8192	1.974 (0.139)	0.23 (0.017)	0.859 (0.056)	0	0	0	MMHC (MMHC)

Child

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	58.797 (6.059)	53.617 (2.156)	53.398 (1.745)	0	0	0.38973808	- (-)
512	43.758 (5.434)	54.07 (4.509)	48.773 (3.215)	0	0	0	GES (LBNA)
2048	39.656 (5.028)	39.094 (4.37)	36.445 (3.071)	0.34748268	0	0	LBNA (LBNA)
8192	34.086 (4.988)	58.18 (3.338)	44.719 (3.992)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.948 (0.085)	0.237 (0.035)	0.238 (0.033)	0	0	0.88875484	GES (-)
512	0.904 (0.088)	0.576 (0.039)	0.425 (0.045)	0	0	0	GES (MMHC)
2048	1.036 (0.062)	0.808 (0.025)	0.691 (0.025)	0	0	0	GES (MMHC)
8192	1.071 (0.042)	0.984 (0.039)	0.639 (0.049)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.576 (0.058)	0.638 (0.111)	0.651 (0.095)	0	0	0.32564259	- (-)
512	0.764 (0.065)	0.708 (0.066)	0.85 (0.064)	0	0	0	LBNA (LBNA)
2048	0.765 (0.046)	0.881 (0.03)	0.984 (0.019)	0	0	0	LBNA (LBNA)
8192	0.803 (0.032)	0.609 (0.036)	0.834 (0.056)	0	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.543 (0.039)	0.151 (0.033)	0.155 (0.031)	0	0	0.34495735	GES (-)
512	0.686 (0.034)	0.407 (0.043)	0.361 (0.042)	0	0	0	GES (MMHC)
2048	0.79 (0.03)	0.711 (0.019)	0.679 (0.023)	0	0	0	GES (MMHC)
8192	0.859 (0.036)	0.598 (0.029)	0.531 (0.033)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.168 (0.068)	0.766 (0.425)	0.66 (0.473)	0	0	0.07241821	- (-)
512	0.378 (0.123)	0.2 (0.143)	0.298 (0.194)	0	8.965e-05	3.81e-06	GES (LBNA)
2048	0.366 (0.121)	0.458 (0.084)	0.617 (0.076)	0	0	0	LBNA (LBNA)
8192	0.439 (0.098)	0.232 (0.068)	0.363 (0.151)	0	4.77e-06	0	GES (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.162 (0.064)	0 (0)	0.001 (0.005)	0	0	0.49808311	GES (-)
512	0.252 (0.087)	0.046 (0.035)	0.052 (0.035)	0	0	0.20489216	GES (-)
2048	0.316 (0.123)	0.275 (0.066)	0.327 (0.083)	0.00108719	0.42713165	0	- (LBNA)
8192	0.543 (0.12)	0.216 (0.039)	0.157 (0.064)	0	0	0	GES (MMHC)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.927 (0.007)	1.208 (0.029)	1.204 (0.031)	0	0	0.30988407	GES (-)
512	0.974 (0.003)	1.153 (0.036)	1.205 (0.043)	0	0	0	GES (MMHC)
2048	0.993 (0.002)	1.045 (0.012)	1.065 (0.012)	0	0	0	GES (MMHC)
8192	0.999 (0.002)	1.273 (0.014)	1.374 (0.036)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.816 (0.016)	0.705 (0.023)	0.707 (0.024)	0	0	0.52572155	GES (-)
512	0.853 (0.011)	0.796 (0.021)	0.778 (0.022)	0	0	0	GES (MMHC)
2048	0.859 (0.01)	0.847 (0.012)	0.844 (0.012)	0	0	0.02025414	GES (MMHC)
8192	0.864 (0.011)	0.76 (0.013)	0.734 (0.017)	0	0	0	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.636 (0.03)	0.059 (0.001)	0.182 (0.021)	0	0	0	MMHC (MMHC)
512	0.85 (0.087)	0.087 (0.006)	0.386 (0.027)	0	0	0	MMHC (MMHC)
2048	1.589 (0.097)	0.158 (0.013)	0.717 (0.038)	0	0	0	MMHC (MMHC)
8192	3.64 (0.257)	0.489 (0.033)	1.956 (0.138)	0	0	0	MMHC (MMHC)

Insurance

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	85.406 (4.822)	65.18 (1.763)	65.383 (1.748)	0	0	0.37465858	- (-)
512	76.375 (5.147)	61.688 (2.184)	61.578 (2.25)	0	0	0.71441936	- (-)
2048	61.047 (4.961)	51.594 (3.915)	52.039 (3.664)	0	0	0.35701275	- (-)
8192	53.82 (4.684)	54.352 (6.372)	51.453 (5.116)	0.45477772	0.00016785	8.678e-05	LBNA (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.824 (0.076)	0.137 (0.025)	0.128 (0.026)	0	0	0.0059309	GES (MMHC)
512	0.769 (0.044)	0.265 (0.025)	0.253 (0.026)	0	0	0.00059605	GES (MMHC)
2048	0.713 (0.034)	0.444 (0.03)	0.384 (0.028)	0	0	0	GES (MMHC)
8192	0.671 (0.032)	0.542 (0.024)	0.425 (0.038)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.456 (0.037)	0.793 (0.136)	0.774 (0.145)	0	0	0.28521919	- (-)
512	0.535 (0.045)	0.909 (0.069)	0.92 (0.074)	0	0	0.20806694	- (-)
2048	0.675 (0.047)	0.912 (0.045)	0.967 (0.037)	0	0	0	LBNA (LBNA)
8192	0.787 (0.043)	0.792 (0.067)	0.9 (0.049)	0.48976898	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.374 (0.034)	0.109 (0.026)	0.099 (0.026)	0	0	0.00289249	GES (MMHC)
512	0.41 (0.03)	0.24 (0.028)	0.233 (0.032)	0	0	0.06279564	GES (-)
2048	0.48 (0.035)	0.405 (0.026)	0.371 (0.027)	0	0	0	GES (MMHC)
8192	0.528 (0.026)	0.429 (0.035)	0.383 (0.042)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.046 (0.037)	1 (0)	1 (0)	0	0	1	- (-)
512	0.13 (0.061)	0.717 (0.439)	0.816 (0.386)	0	0	0.06094837	- (-)
2048	0.351 (0.087)	0.768 (0.178)	0.852 (0.172)	0	0	0.00016212	LBNA (LBNA)
8192	0.511 (0.118)	0.587 (0.196)	0.75 (0.164)	0.00017738	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.02 (0.016)	0 (0)	0 (0)	0	0	1	GES (-)
512	0.049 (0.024)	0.003 (0.006)	0.004 (0.006)	0	0	1	GES (-)
2048	0.112 (0.028)	0.062 (0.023)	0.052 (0.021)	0	0	0.00025654	GES (MMHC)
8192	0.131 (0.034)	0.091 (0.029)	0.071 (0.025)	0	0	0	GES (MMHC)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.94 (0.012)	1.103 (0.025)	1.11 (0.024)	0	0	0.01997948	GES (MMHC)
512	0.967 (0.003)	1.041 (0.025)	1.048 (0.029)	0	0	0.03447056	GES (MMHC)
2048	0.988 (0.001)	0.993 (0.004)	0.995 (0.003)	0	0	0	GES (MMHC)
8192	0.997 (0)	1.097 (0.042)	1.104 (0.042)	0	0	0.14866543	GES (-)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.822 (0.014)	0.797 (0.017)	0.794 (0.019)	0	0	0.28186417	GES (-)
512	0.844 (0.011)	0.831 (0.012)	0.828 (0.016)	0	0	0.17302132	GES (-)
2048	0.849 (0.012)	0.85 (0.012)	0.848 (0.012)	0.63027191	0.31819153	0.13140011	- (-)
8192	0.85 (0.01)	0.822 (0.018)	0.821 (0.019)	0	0	0.81951714	GES (-)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.775 (0.11)	0.071 (0.004)	0.164 (0.019)	0	0	0	MMHC (MMHC)
512	1.251 (0.105)	0.085 (0.004)	0.254 (0.02)	0	0	0	MMHC (MMHC)
2048	1.606 (0.109)	0.12 (0.009)	0.473 (0.035)	0	0	0	MMHC (MMHC)
8192	3.611 (0.38)	0.247 (0.016)	0.978 (0.062)	0	0	0	MMHC (MMHC)

Water

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	51.844 (2.958)	46.609 (0.755)	46.586 (0.737)	0	0	0.86689091	- (-)
512	38.242 (4.715)	47.805 (1.137)	47.758 (1.215)	0	0	0.79033566	GES (-)
2048	26.844 (1.239)	48.906 (0.9)	47.766 (0.715)	0	0	0	GES (LBNA)
8192	22.539 (1.474)	50.75 (1.087)	48.875 (1.431)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.575 (0.043)	0.047 (0.027)	0.046 (0.026)	0	0	0.63787556	GES (-)
512	0.61 (0.012)	0.188 (0.035)	0.188 (0.037)	0	0	1	GES (-)
2048	0.69 (0.014)	0.23 (0.031)	0.195 (0.026)	0	0	0	GES (MMHC)
8192	0.796 (0.02)	0.366 (0.037)	0.322 (0.045)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.717 (0.06)	0.757 (0.303)	0.766 (0.304)	0.1408205	0.07133102	0.8111105	- (-)
512	0.867 (0.025)	0.797 (0.12)	0.801 (0.126)	0	0	0.78454113	GES (-)
2048	0.924 (0.018)	0.728 (0.07)	0.801 (0.08)	0	0	0	GES (LBNA)
8192	0.928 (0.022)	0.683 (0.053)	0.758 (0.079)	0	0	0	GES (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.41 (0.029)	0.033 (0.021)	0.032 (0.02)	0	0	0.7908535	GES (-)
512	0.529 (0.016)	0.148 (0.03)	0.149 (0.032)	0	0	0.86144543	GES (-)
2048	0.637 (0.017)	0.167 (0.024)	0.156 (0.027)	0	0	0.00070286	GES (MMHC)
8192	0.738 (0.016)	0.25 (0.031)	0.243 (0.033)	0	0	0.06872749	GES (-)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.875 (0.314)	1 (0)	1 (0)	9.5e-07	9.5e-07	1	- (-)
512	0.995 (0.037)	1 (0)	1 (0)	0.49903774	0.49699879	1	- (-)
2048	0.972 (0.048)	1 (0)	1 (0)	0	0	1	- (-)
8192	0.976 (0.041)	0.32 (0.468)	0.402 (0.482)	0	0	0.18923759	GES (-)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.011 (0.016)	0 (0)	0 (0)	0	0	1	GES (-)
512	0.075 (0.024)	0 (0)	0 (0.002)	0	0	1	GES (-)
2048	0.234 (0.016)	0 (0)	0 (0.002)	0	0	1	GES (-)
8192	0.322 (0.016)	0 (0)	0.001 (0.006)	0	0	0.02940369	GES (LBNA)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.858 (0.009)	0.958 (0.013)	0.956 (0.012)	0	0	0.16080093	GES (-)
512	0.848 (0.004)	0.999 (0.008)	0.998 (0.007)	0	0	0.76133156	GES (-)
2048	0.903 (0.002)	1.121 (0.007)	1.132 (0.005)	0	0	0	GES (MMHC)
8192	0.982 (0.001)	1.251 (0.011)	1.255 (0.013)	0	0	0.00836182	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.477 (0.015)	0.332 (0.017)	0.332 (0.016)	0	0	0.97903824	GES (-)
512	0.524 (0.016)	0.373 (0.018)	0.373 (0.016)	0	0	0.89291668	GES (-)
2048	0.568 (0.015)	0.383 (0.017)	0.381 (0.018)	0	0	0.36129379	GES (-)
8192	0.583 (0.017)	0.428 (0.02)	0.424 (0.021)	0	0	0.14441776	GES (-)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.994 (0.105)	0.072 (0.004)	0.106 (0.014)	0	0	0	MMHC (MMHC)
512	1.711 (0.19)	0.084 (0.006)	0.181 (0.018)	0	0	0	MMHC (MMHC)
2048	4.085 (0.317)	0.117 (0.002)	0.24 (0.015)	0	0	0	MMHC (MMHC)
8192	6.608 (0.381)	0.251 (0.012)	0.55 (0.039)	0	0	0	MMHC (MMHC)

Mildew

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	70.422 (8.01)	48.805 (3.448)	49.039 (3.002)	0	0	0.57493782	- (-)
512	50.805 (5.147)	32.344 (4.979)	28.773 (4.437)	0	0	0	LBNA (LBNA)
2048	35.711 (5.466)	20.641 (4.35)	8.219 (2.948)	0	0	0	LBNA (LBNA)
8192	23.219 (7.036)	23.594 (4.048)	13.023 (3.568)	0.60991859	0	0	LBNA (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	2.021 (0.11)	0.464 (0.051)	0.429 (0.055)	0	0	0	GES (MMHC)
512	1.775 (0.073)	0.944 (0.047)	0.733 (0.054)	0	0	0	GES (MMHC)
2048	1.507 (0.064)	1.043 (0.029)	0.947 (0.024)	0	0	0	GES (MMHC)
8192	1.286 (0.062)	1.024 (0.038)	0.898 (0.035)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.412 (0.033)	0.604 (0.082)	0.612 (0.082)	0	0	0.40297413	- (-)
512	0.524 (0.026)	0.783 (0.04)	0.878 (0.043)	0	0	0	LBNA (LBNA)
2048	0.631 (0.03)	0.878 (0.029)	0.968 (0.02)	0	0	0	LBNA (LBNA)
8192	0.755 (0.038)	0.865 (0.035)	0.969 (0.026)	0	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.829 (0.049)	0.28 (0.048)	0.263 (0.048)	0	0	0.00478554	GES (MMHC)
512	0.929 (0.023)	0.738 (0.036)	0.642 (0.046)	0	0	0	GES (MMHC)
2048	0.948 (0.014)	0.916 (0.017)	0.916 (0.014)	0	0	0.79919052	GES (-)
8192	0.969 (0.014)	0.884 (0.019)	0.869 (0.024)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.15 (0.038)	0.34 (0.245)	0.294 (0.309)	0	0	0.18533802	- (-)
512	0.283 (0.051)	0.597 (0.132)	0.775 (0.133)	0	0	0	LBNA (LBNA)
2048	0.471 (0.082)	0.825 (0.086)	0.89 (0.077)	0	0	0	LBNA (LBNA)
8192	0.66 (0.137)	0.726 (0.091)	0.909 (0.103)	5.72e-06	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.466 (0.105)	0.047 (0.037)	0.033 (0.036)	0	0	0.0025177	GES (MMHC)
512	0.597 (0.086)	0.362 (0.083)	0.351 (0.081)	0	0	0.29449272	GES (-)
2048	0.672 (0.104)	0.577 (0.05)	0.662 (0.04)	0	0.37257671	0	- (LBNA)
8192	0.736 (0.123)	0.474 (0.064)	0.57 (0.062)	0	0	0	GES (LBNA)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.942 (0.009)	1.416 (0.034)	1.426 (0.043)	0	0	0.0325613	GES (MMHC)
512	0.988 (0.003)	1.142 (0.025)	1.263 (0.041)	0	0	0	GES (MMHC)
2048	1 (0.002)	1.007 (0.006)	1.019 (0.013)	0	0	0	GES (MMHC)
8192	1.001 (0.001)	1.081 (0.019)	1.106 (0.035)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.896 (0.013)	0.86 (0.014)	0.857 (0.015)	0	0	0.18245602	GES (-)
512	0.935 (0.008)	0.929 (0.01)	0.91 (0.012)	0	0	0	GES (MMHC)
2048	0.943 (0.007)	0.944 (0.007)	0.942 (0.007)	0.69080925	0.29078484	0.14300728	- (-)
8192	0.946 (0.007)	0.935 (0.008)	0.935 (0.01)	0	0	0.93050194	GES (-)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.963 (0.058)	0.096 (0.007)	0.306 (0.027)	0	0	0	MMHC (MMHC)
512	1.228 (0.083)	0.133 (0.003)	0.61 (0.042)	0	0	0	MMHC (MMHC)
2048	1.885 (0.125)	0.198 (0.012)	0.878 (0.035)	0	0	0	MMHC (MMHC)
8192	4.338 (0.197)	0.418 (0.021)	1.46 (0.077)	0	0	0	MMHC (MMHC)

Alarm

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	73.984 (4.699)	84.344 (0.524)	84.312 (0.558)	0	0	0.72964096	GES (-)
512	67.5 (5.547)	85.727 (0.801)	85.695 (0.865)	0	0	0.82225323	GES (-)
2048	62.836 (5.719)	85.875 (1.612)	85.094 (1.554)	0	0	0.00010014	GES (LBNA)
8192	46.219 (4.7)	86.398 (4.022)	80.062 (3.884)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.463 (0.038)	0.007 (0.009)	0.007 (0.008)	0	0	0.62022686	GES (-)
512	0.593 (0.056)	0.066 (0.017)	0.064 (0.016)	0	0	0.49463272	GES (-)
2048	0.761 (0.07)	0.226 (0.022)	0.203 (0.02)	0	0	0	GES (MMHC)
8192	0.912 (0.05)	0.459 (0.031)	0.349 (0.029)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.756 (0.059)	0.738 (0.406)	0.737 (0.422)	0.62404633	0.61211586	1	- (-)
512	0.803 (0.067)	0.661 (0.134)	0.67 (0.155)	0	0	0.5877676	GES (-)
2048	0.81 (0.071)	0.75 (0.048)	0.792 (0.053)	0	0.02175617	0	GES (LBNA)
8192	0.809 (0.04)	0.604 (0.055)	0.713 (0.071)	0	0	0	GES (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.349 (0.025)	0.004 (0.006)	0.004 (0.006)	0	0	0.89540386	GES (-)
512	0.473 (0.024)	0.044 (0.015)	0.043 (0.014)	0	0	0.80305576	GES (-)
2048	0.611 (0.017)	0.169 (0.017)	0.161 (0.018)	0	0	0.00017738	GES (MMHC)
8192	0.736 (0.015)	0.277 (0.03)	0.248 (0.024)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.699 (0.137)	1 (0)	1 (0)	0	0	1	- (-)
512	0.757 (0.16)	1 (0)	1 (0)	0	0	1	- (-)
2048	0.605 (0.22)	0.863 (0.324)	1 (0)	0	0	9.5e-07	LBNA (LBNA)
8192	0.626 (0.119)	0.78 (0.164)	0.941 (0.104)	0	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.131 (0.03)	0 (0)	0 (0)	0	0	1	GES (-)
512	0.204 (0.018)	0 (0)	0 (0)	0	0	1	GES (-)
2048	0.253 (0.015)	0.006 (0.009)	0.005 (0.008)	0	0	0.32639503	GES (-)
8192	0.504 (0.035)	0.078 (0.02)	0.079 (0.019)	0	0	0.74497795	GES (-)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.861 (0.006)	1.041 (0.007)	1.042 (0.007)	0	0	0.42513752	GES (-)
512	0.843 (0.005)	1.112 (0.013)	1.113 (0.013)	0	0	0.55907249	GES (-)
2048	0.856 (0.003)	1.2 (0.014)	1.203 (0.014)	0	0	0.09266186	GES (-)
8192	0.894 (0.001)	1.278 (0.024)	1.295 (0.019)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.61 (0.018)	0.323 (0.017)	0.32 (0.015)	0	0	0.22045135	GES (-)
512	0.688 (0.016)	0.37 (0.023)	0.368 (0.024)	0	0	0.51711655	GES (-)
2048	0.719 (0.016)	0.421 (0.022)	0.415 (0.022)	0	0	0.03092384	GES (MMHC)
8192	0.747 (0.013)	0.462 (0.029)	0.45 (0.025)	0	0	0.00062561	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.031 (0.137)	0.116 (0.006)	0.134 (0.014)	0	0	0	MMHC (MMHC)
512	1.952 (0.234)	0.143 (0.008)	0.214 (0.016)	0	0	0	MMHC (MMHC)
2048	3.975 (0.608)	0.206 (0.011)	0.414 (0.027)	0	0	0	MMHC (MMHC)
8192	16.003 (2.09)	0.471 (0.034)	1.154 (0.066)	0	0	0	MMHC (MMHC)

Barley

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	73.133 (4.962)	67.609 (1.393)	67.836 (1.391)	0	0	0.21018696	- (-)
512	52.609 (5.626)	62.812 (3.484)	61.422 (3.106)	0	0	0.00093174	GES (LBNA)
2048	42.641 (5.936)	51.773 (4.252)	46.062 (2.924)	0	0	0	GES (LBNA)
8192	42.156 (5.066)	59.383 (4.928)	51.047 (4.517)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.995 (0.036)	0.157 (0.031)	0.157 (0.029)	0	0	0.92421627	GES (-)
512	0.991 (0.025)	0.55 (0.041)	0.495 (0.033)	0	0	0	GES (MMHC)
2048	1 (0.02)	0.787 (0.052)	0.66 (0.036)	0	0	0	GES (MMHC)
8192	1.089 (0.017)	0.918 (0.044)	0.73 (0.042)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.57 (0.041)	0.789 (0.099)	0.771 (0.099)	0	0	0.15363026	- (-)
512	0.738 (0.047)	0.766 (0.046)	0.8 (0.042)	0	0	0	LBNA (LBNA)
2048	0.779 (0.055)	0.683 (0.052)	0.78 (0.04)	0	0.86693192	0	- (LBNA)
8192	0.76 (0.07)	0.587 (0.039)	0.689 (0.049)	0	0	0	GES (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.567 (0.039)	0.123 (0.024)	0.12 (0.022)	0	0	0.43474865	GES (-)
512	0.73 (0.04)	0.42 (0.02)	0.395 (0.022)	0	0	0	GES (MMHC)
2048	0.778 (0.052)	0.536 (0.028)	0.513 (0.026)	0	0	0	GES (MMHC)
8192	0.828 (0.077)	0.538 (0.033)	0.502 (0.031)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.464 (0.095)	0.984 (0.125)	1 (0)	0	0	0.49807072	- (-)
512	0.669 (0.053)	0.957 (0.08)	0.977 (0.064)	0	0	0.02759171	LBNA (LBNA)
2048	0.728 (0.065)	0.825 (0.12)	0.932 (0.081)	0	0	0	LBNA (LBNA)
8192	0.706 (0.025)	0.672 (0.092)	0.776 (0.087)	9.441e-05	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.149 (0.035)	0 (0)	0 (0)	0	0	1	GES (-)
512	0.27 (0.024)	0.145 (0.035)	0.14 (0.033)	0	0	0.24977875	GES (-)
2048	0.416 (0.043)	0.373 (0.031)	0.366 (0.029)	0	0	0.07379532	GES (-)
8192	0.551 (0.022)	0.351 (0.034)	0.35 (0.037)	0	0	0.8865881	GES (-)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.947 (0.004)	1.152 (0.013)	1.151 (0.012)	0	0	0.48352623	GES (-)
512	0.963 (0.002)	1.156 (0.017)	1.162 (0.016)	0	0	0.00233841	GES (MMHC)
2048	0.98 (0.001)	1.162 (0.016)	1.177 (0.019)	0	0	0	GES (MMHC)
8192	0.992 (0)	1.207 (0.014)	1.224 (0.013)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.64 (0.017)	0.45 (0.021)	0.453 (0.023)	0	0	0.32006931	GES (-)
512	0.68 (0.015)	0.552 (0.021)	0.547 (0.021)	0	0	0.04781723	GES (MMHC)
2048	0.693 (0.016)	0.594 (0.019)	0.583 (0.022)	0	0	2.67e-05	GES (MMHC)
8192	0.695 (0.014)	0.59 (0.022)	0.576 (0.019)	0	0	0	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.237 (0.149)	0.167 (0.008)	0.281 (0.019)	0	0	0	MMHC (MMHC)
512	2.014 (0.248)	0.214 (0.007)	0.636 (0.037)	0	0	0	MMHC (MMHC)
2048	3.647 (0.287)	0.322 (0.018)	1.094 (0.062)	0	0	0	MMHC (MMHC)
8192	9.796 (0.787)	0.802 (0.039)	2.159 (0.122)	0	0	0	MMHC (MMHC)

Hailfinder

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	267.297 (10.517)	144.352 (5.384)	141.703 (4.622)	0	0	3.815e-05	LBNA (LBNA)
512	185.594 (9.903)	115.641 (6.044)	114.008 (6.16)	0	0	0.03449345	LBNA (LBNA)
2048	123.43 (6.451)	83.008 (4.861)	82.023 (4.468)	0	0	0.09598637	- (-)
8192	77.961 (6.798)	55.688 (4.557)	54.758 (4.383)	0	0	0.10015202	- (-)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.631 (0.077)	0.363 (0.038)	0.339 (0.03)	0	0	0	GES (MMHC)
512	1.26 (0.06)	0.5 (0.033)	0.476 (0.032)	0	0	0	GES (MMHC)
2048	1.089 (0.04)	0.623 (0.028)	0.601 (0.03)	0	0	0	GES (MMHC)
8192	1.029 (0.034)	0.755 (0.024)	0.719 (0.024)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.176 (0.018)	0.374 (0.06)	0.398 (0.06)	0	0	0.00122547	LBNA (LBNA)
512	0.344 (0.027)	0.668 (0.045)	0.691 (0.049)	0	0	0.00011826	LBNA (LBNA)
2048	0.558 (0.025)	0.848 (0.037)	0.869 (0.036)	0	0	1.91e-06	LBNA (LBNA)
8192	0.742 (0.024)	0.916 (0.024)	0.948 (0.022)	0	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.288 (0.029)	0.135 (0.023)	0.134 (0.019)	0	0	0.73647499	GES (-)
512	0.433 (0.03)	0.333 (0.022)	0.328 (0.022)	0	0	0.08736706	GES (-)
2048	0.608 (0.021)	0.528 (0.02)	0.522 (0.02)	0	0	0.02071953	GES (MMHC)
8192	0.763 (0.017)	0.691 (0.017)	0.681 (0.017)	0	0	9.54e-06	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.009 (0.006)	0.144 (0.143)	0.169 (0.147)	0	0	0.16171169	- (-)
512	0.066 (0.021)	0.408 (0.136)	0.421 (0.135)	0	0	0.46645069	- (-)
2048	0.213 (0.035)	0.685 (0.101)	0.682 (0.104)	0	0	0.82790661	- (-)
8192	0.465 (0.05)	0.786 (0.052)	0.821 (0.063)	0	0	1.91e-06	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.021 (0.013)	0.011 (0.009)	0.01 (0.007)	0	0	0.30868626	GES (-)
512	0.078 (0.023)	0.044 (0.015)	0.042 (0.013)	0	0	0.51564693	GES (-)
2048	0.168 (0.024)	0.128 (0.021)	0.125 (0.021)	0	0	0.21750164	GES (-)
8192	0.33 (0.037)	0.301 (0.029)	0.295 (0.034)	0	0	0.135849	GES (-)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.903 (0.006)	0.975 (0.007)	0.975 (0.007)	0	0	0.40281677	GES (-)
512	0.969 (0.002)	0.98 (0.002)	0.98 (0.002)	0	0	0.27415848	GES (-)
2048	0.99 (0.001)	0.992 (0.001)	0.992 (0.001)	0	0	0	GES (MMHC)
8192	0.997 (0)	1.003 (0.001)	1.003 (0.002)	0	0	9.441e-05	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.709 (0.018)	0.786 (0.013)	0.787 (0.014)	0	0	0.58086395	- (-)
512	0.793 (0.012)	0.803 (0.013)	0.803 (0.012)	0	0	0.78430748	- (-)
2048	0.806 (0.013)	0.806 (0.012)	0.807 (0.013)	0.9196682	0.33240414	0.37618351	- (-)
8192	0.809 (0.011)	0.805 (0.012)	0.801 (0.014)	0.0095768	2.86e-06	0.01872444	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	2.345 (0.325)	0.28 (0.011)	0.776 (0.054)	0	0	0	MMHC (MMHC)
512	2.853 (0.216)	0.322 (0.014)	1.113 (0.068)	0	0	0	MMHC (MMHC)
2048	5.299 (0.411)	0.433 (0.019)	1.696 (0.086)	0	0	0	MMHC (MMHC)
8192	15.77 (1.032)	1.107 (0.087)	3.66 (0.16)	0	0	0	MMHC (MMHC)

Hepar II

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	287.086 (16.962)	97.719 (6.961)	98.445 (6.331)	0	0	0.38880253	- (-)
512	268.875 (14.601)	77.062 (6.06)	79.211 (6.283)	0	0	0.00603485	MMHC (MMHC)
2048	234.07 (9.595)	64.492 (8.16)	62.398 (5.291)	0	0	0.01582813	LBNA (LBNA)
8192	213.547 (7.826)	57.32 (8.306)	49.625 (6.372)	0	0	0	LBNA (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	2.567 (0.169)	0.489 (0.042)	0.414 (0.036)	0	0	0	GES (MMHC)
512	2.752 (0.103)	0.748 (0.043)	0.586 (0.031)	0	0	0	GES (MMHC)
2048	2.639 (0.057)	0.935 (0.042)	0.742 (0.036)	0	0	0	GES (MMHC)
8192	2.519 (0.066)	1.05 (0.043)	0.85 (0.03)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.225 (0.017)	0.741 (0.057)	0.778 (0.061)	0	0	9.5e-07	LBNA (LBNA)
512	0.27 (0.017)	0.774 (0.036)	0.837 (0.043)	0	0	0	LBNA (LBNA)
2048	0.319 (0.013)	0.771 (0.034)	0.858 (0.028)	0	0	0	LBNA (LBNA)
8192	0.348 (0.01)	0.771 (0.036)	0.88 (0.026)	0	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.577 (0.05)	0.362 (0.035)	0.321 (0.029)	0	0	0	GES (MMHC)
512	0.741 (0.037)	0.578 (0.026)	0.49 (0.027)	0	0	0	GES (MMHC)
2048	0.84 (0.031)	0.72 (0.029)	0.636 (0.027)	0	0	0	GES (MMHC)
8192	0.876 (0.019)	0.809 (0.025)	0.748 (0.027)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.073 (0.02)	0.647 (0.145)	0.654 (0.153)	0	0	0.69601536	- (-)
512	0.127 (0.023)	0.662 (0.076)	0.696 (0.093)	0	0	0.00122356	LBNA (LBNA)
2048	0.185 (0.023)	0.64 (0.066)	0.715 (0.063)	0	0	0	LBNA (LBNA)
8192	0.239 (0.02)	0.659 (0.068)	0.776 (0.064)	0	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.261 (0.071)	0.096 (0.031)	0.064 (0.022)	0	0	0	GES (MMHC)
512	0.475 (0.078)	0.274 (0.046)	0.161 (0.031)	0	0	0	GES (MMHC)
2048	0.64 (0.066)	0.49 (0.063)	0.316 (0.04)	0	0	0	GES (MMHC)
8192	0.688 (0.04)	0.65 (0.06)	0.503 (0.063)	0	0	0	GES (MMHC)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.823 (0.021)	1.203 (0.031)	1.252 (0.029)	0	0	0	GES (MMHC)
512	0.919 (0.006)	1.106 (0.026)	1.205 (0.03)	0	0	0	GES (MMHC)
2048	0.979 (0.003)	1.057 (0.018)	1.149 (0.032)	0	0	0	GES (MMHC)
8192	0.995 (0.001)	1.129 (0.028)	1.177 (0.027)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.955 (0.007)	0.958 (0.008)	0.957 (0.007)	0.00462818	0.02321148	0.53817654	- (-)
512	0.971 (0.005)	0.971 (0.006)	0.967 (0.006)	0.55114555	0	6.68e-06	- (MMHC)
2048	0.977 (0.005)	0.976 (0.005)	0.974 (0.005)	0.02208519	0	0.0006609	GES (MMHC)
8192	0.979 (0.005)	0.976 (0.005)	0.974 (0.006)	0	0	0.00378704	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	4.931 (5.566)	0.33 (0.012)	1.139 (1.51)	0	0	0	MMHC (MMHC)
512	6.462 (1.346)	0.39 (0.013)	1.772 (0.127)	0	0	0	MMHC (MMHC)
2048	10.955 (1.111)	0.546 (0.025)	3.123 (0.222)	0	0	0	MMHC (MMHC)
8192	34.38 (4.825)	1.258 (0.175)	5.767 (0.465)	0	0	0	MMHC (MMHC)

Win95pts

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	217.867 (4.64)	214.781 (4.008)	0	LBNA
512	283.961 (8.292)	237.445 (5.76)	0	LBNA
2048	374.148 (7.122)	270.758 (6.201)	0	LBNA
8192	380.664 (12.326)	253.977 (7.847)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.18 (0.022)	0.166 (0.019)	0	MMHC
512	0.588 (0.04)	0.324 (0.029)	0	MMHC
2048	1.141 (0.034)	0.542 (0.031)	0	MMHC
8192	1.311 (0.041)	0.548 (0.029)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.224 (0.053)	0.254 (0.057)	1.621e-05	LBNA
512	0.156 (0.021)	0.219 (0.037)	0	LBNA
2048	0.136 (0.011)	0.206 (0.021)	0	LBNA
8192	0.19 (0.01)	0.339 (0.024)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.04 (0.01)	0.042 (0.009)	0.13699627	-
512	0.091 (0.011)	0.071 (0.013)	0	MMHC
2048	0.155 (0.013)	0.111 (0.012)	0	MMHC
8192	0.249 (0.01)	0.185 (0.011)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.078 (0.269)	0.234 (0.425)	0.00086212	LBNA
512	0.002 (0.007)	0.002 (0.011)	0.77571583	-
2048	0.003 (0.004)	0.016 (0.012)	0	LBNA
8192	0.002 (0.003)	0.006 (0.013)	0.00147915	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0 (0)	0 (0)	1	-
512	0.004 (0.013)	0.001 (0.008)	0.08385849	-
2048	0.025 (0.027)	0.034 (0.024)	0.01072693	LBNA
8192	0.029 (0.034)	0.015 (0.032)	0.00082111	MMHC

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.927 (0.031)	0.93 (0.032)	0.4529686	-
512	1.072 (0.024)	1.186 (0.029)	0	MMHC
2048	1.178 (0.027)	1.564 (0.036)	0	MMHC
8192	1.391 (0.058)	1.99 (0.037)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.809 (0.014)	0.805 (0.016)	0.0348711	MMHC
512	0.862 (0.013)	0.829 (0.016)	0	MMHC
2048	0.879 (0.012)	0.826 (0.015)	0	MMHC
8192	0.887 (0.011)	0.816 (0.014)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.64 (0.614)	1.055 (0.09)	0	MMHC
512	0.829 (0.031)	2.53 (0.188)	0	MMHC
2048	2.583 (0.151)	10.426 (1.17)	0	MMHC
8192	13.929 (1.011)	45.956 (4.158)	0	MMHC

Pathfinder

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	304.023 (4.05)	303.008 (4.277)	0.05402946	-
512	346.625 (7.503)	324 (5.274)	0	LBNA
2048	340.203 (12.771)	288.875 (6.788)	0	LBNA
8192	419.125 (10.053)	276.273 (7.275)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.153 (0.017)	0.148 (0.016)	0.03245258	MMHC
512	0.404 (0.024)	0.287 (0.017)	0	MMHC
2048	0.759 (0.027)	0.339 (0.023)	0	MMHC
8192	1.127 (0.023)	0.41 (0.023)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.212 (0.048)	0.223 (0.046)	0.05502319	-
512	0.244 (0.03)	0.283 (0.039)	0	LBNA
2048	0.388 (0.028)	0.609 (0.043)	0	LBNA
8192	0.305 (0.012)	0.632 (0.033)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.032 (0.008)	0.033 (0.007)	0.63233757	-
512	0.098 (0.011)	0.081 (0.012)	0	MMHC
2048	0.294 (0.019)	0.206 (0.019)	0	MMHC
8192	0.344 (0.013)	0.259 (0.015)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.458 (0.477)	0.568 (0.486)	0.07080173	-
512	0.114 (0.077)	0.324 (0.215)	0	LBNA
2048	0.163 (0.044)	0.611 (0.131)	0	LBNA
8192	0.088 (0.016)	0.465 (0.095)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.002 (0.003)	0.002 (0.003)	1	-
512	0.011 (0.007)	0.01 (0.006)	0.07587624	-
2048	0.092 (0.022)	0.057 (0.014)	0	MMHC
8192	0.149 (0.025)	0.102 (0.02)	0	MMHC

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.768 (0.029)	1.761 (0.033)	0.09492016	-
512	1.83 (0.03)	1.907 (0.033)	0	MMHC
2048	1.553 (0.036)	1.942 (0.045)	0	MMHC
8192	1.549 (0.028)	2.233 (0.029)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.845 (0.011)	0.845 (0.012)	0.73366165	-
512	0.866 (0.012)	0.862 (0.013)	0.03624153	MMHC
2048	0.899 (0.01)	0.878 (0.011)	0	MMHC
8192	0.894 (0.009)	0.851 (0.012)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.698 (0.47)	2.475 (0.76)	0	MMHC
512	2.045 (0.033)	3.804 (0.12)	0	MMHC
2048	3.499 (0.09)	7.664 (0.324)	0	MMHC
8192	19.133 (2.152)	37.804 (2.815)	0	MMHC

Munin1

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	993.211 (28.438)	365.57 (16.151)	343.32 (13.801)	0	0	0	LBNA (LBNA)
512	593.562 (17.631)	256.383 (12.867)	237.461 (10.874)	0	0	0	LBNA (LBNA)
2048	398.672 (12.073)	171.852 (9.483)	157.961 (7.681)	0	0	0	LBNA (LBNA)
8192	250.75 (13.026)	132.555 (11.158)	115.805 (8.191)	0	0	0	LBNA (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	2.983 (0.073)	0.894 (0.036)	0.757 (0.028)	0	0	0	GES (MMHC)
512	2.208 (0.04)	1.01 (0.029)	0.91 (0.024)	0	0	0	GES (MMHC)
2048	1.845 (0.031)	1.028 (0.022)	0.949 (0.019)	0	0	0	GES (MMHC)
8192	1.585 (0.028)	1.077 (0.022)	0.992 (0.017)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.189 (0.008)	0.491 (0.025)	0.54 (0.027)	0	0	0	LBNA (LBNA)
512	0.339 (0.01)	0.633 (0.021)	0.678 (0.018)	0	0	0	LBNA (LBNA)
2048	0.461 (0.009)	0.747 (0.016)	0.789 (0.015)	0	0	0	LBNA (LBNA)
8192	0.592 (0.013)	0.791 (0.018)	0.841 (0.015)	0	0	0	LBNA (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.564 (0.02)	0.438 (0.017)	0.408 (0.017)	0	0	0	GES (MMHC)
512	0.749 (0.018)	0.639 (0.014)	0.617 (0.013)	0	0	0	GES (MMHC)
2048	0.85 (0.011)	0.768 (0.013)	0.749 (0.011)	0	0	0	GES (MMHC)
8192	0.939 (0.016)	0.851 (0.012)	0.834 (0.011)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.041 (0.005)	0.343 (0.045)	0.419 (0.051)	0	0	0	LBNA (LBNA)
512	0.16 (0.009)	0.599 (0.04)	0.684 (0.043)	0	0	0	LBNA (LBNA)
2048	0.302 (0.012)	0.785 (0.034)	0.868 (0.029)	0	0	0	LBNA (LBNA)
8192	0.474 (0.023)	0.806 (0.038)	0.913 (0.021)	0	0	0	LBNA (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.262 (0.023)	0.149 (0.015)	0.133 (0.013)	0	0	0	GES (MMHC)
512	0.5 (0.024)	0.342 (0.017)	0.32 (0.019)	0	0	0	GES (MMHC)
2048	0.661 (0.017)	0.524 (0.022)	0.504 (0.019)	0	0	0	GES (MMHC)
8192	0.836 (0.036)	0.666 (0.023)	0.647 (0.02)	0	0	0	GES (MMHC)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.878 (0.005)	1.005 (0.005)	1.019 (0.005)	0	0	0	GES (MMHC)
512	0.976 (0.001)	1.014 (0.003)	1.019 (0.003)	0	0	0	GES (MMHC)
2048	0.995 (0)	1.015 (0.002)	1.021 (0.002)	0	0	0	GES (MMHC)
8192	0.999 (0)	1.02 (0.004)	1.026 (0.004)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.699 (0.014)	0.795 (0.013)	0.795 (0.012)	0	0	0.75503635	- (-)
512	0.822 (0.012)	0.828 (0.013)	0.829 (0.011)	6.485e-05	4.77e-06	0.67952156	- (-)
2048	0.84 (0.009)	0.837 (0.012)	0.836 (0.01)	0.01904678	0.00058079	0.43875885	GES (-)
8192	0.842 (0.012)	0.836 (0.011)	0.836 (0.012)	0.0001545	6.58e-05	0.75481796	GES (-)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	56.362 (96.287)	3.129 (2.261)	7.883 (0.912)	0	0	0	MMHC (MMHC)
512	27.74 (1.976)	3.151 (0.04)	10 (0.32)	0	0	0	MMHC (MMHC)
2048	57.497 (2.838)	3.843 (0.042)	12.392 (0.321)	0	0	0	MMHC (MMHC)
8192	184.927 (11.306)	12.183 (12.258)	25.847 (9.211)	0	0	0	MMHC (MMHC)

Andes

Structural Hamming Distance							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	741.93 (29.947)	606.703 (1.816)	606.852 (2.012)	0	0	0.55788708	- (-)
512	653.109 (19.858)	622.531 (5.306)	620.859 (3.828)	0	0	0.00428963	LBNA (LBNA)
2048	476.461 (19.286)	850.648 (10.54)	696.617 (9.419)	0	0	0	GES (LBNA)
8192	381.672 (6.015)	1067.055 (13.354)	714.375 (10.45)	0	0	0	GES (LBNA)

Normalized number of edges							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.217 (0.035)	0.033 (0.003)	0.033 (0.003)	0	0	0.74897194	GES (-)
512	1.237 (0.012)	0.05 (0.011)	0.048 (0.008)	0	0	0.16172504	GES (-)
2048	1.236 (0.016)	0.492 (0.018)	0.227 (0.015)	0	0	0	GES (MMHC)
8192	1.182 (0.006)	0.995 (0.024)	0.307 (0.017)	0	0	0	GES (MMHC)

Skeleton precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.498 (0.017)	0.764 (0.088)	0.758 (0.097)	0	0	0.60716534	- (-)
512	0.561 (0.01)	0.255 (0.076)	0.282 (0.073)	0	0	0.00383186	GES (LBNA)
2048	0.65 (0.014)	0.14 (0.009)	0.294 (0.03)	0	0	0	GES (LBNA)
8192	0.722 (0.005)	0.144 (0.009)	0.374 (0.028)	0	0	0	GES (LBNA)

Skeleton recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.606 (0.011)	0.025 (0.004)	0.025 (0.004)	0	0	0.85316944	GES (-)
512	0.693 (0.009)	0.013 (0.005)	0.014 (0.004)	0	0	0.10390091	GES (-)
2048	0.803 (0.01)	0.069 (0.005)	0.066 (0.005)	0	0	0.00030327	GES (MMHC)
8192	0.854 (0.004)	0.143 (0.007)	0.115 (0.007)	0	0	0	GES (MMHC)

V-structures precision							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.2 (0.019)	1 (0)	1 (0)	0	0	1	- (-)
512	0.288 (0.017)	0.359 (0.478)	0.492 (0.498)	0.09175682	8.58e-06	0.03584671	LBNA (LBNA)
2048	0.35 (0.022)	0 (0)	0 (0)	0	0	1	GES (-)
8192	0.442 (0.015)	0.005 (0.002)	0.052 (0.031)	0	0	0	GES (LBNA)

V-structures recall							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.329 (0.016)	0 (0)	0 (0)	0	0	1	GES (-)
512	0.426 (0.015)	0 (0.001)	0 (0.001)	0	0	1	GES (-)
2048	0.468 (0.011)	0 (0)	0 (0)	0	0	1	GES (-)
8192	0.593 (0.002)	0.005 (0.002)	0.004 (0.002)	0	0	0.27771568	GES (-)

Normalized score							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	1.047 (0.006)	1.71 (0.007)	1.709 (0.007)	0	0	0.50256443	GES (-)
512	1.07 (0.004)	2.421 (0.011)	2.419 (0.01)	0	0	0.05605602	GES (-)
2048	1.065 (0.016)	2.869 (0.012)	2.935 (0.013)	0	0	0	GES (MMHC)
8192	1.02 (0.004)	3.144 (0.018)	3.446 (0.018)	0	0	0	GES (MMHC)

Prediction accuracy							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	0.835 (0.014)	0.353 (0.015)	0.355 (0.015)	0	0	0.21954346	GES (-)
512	0.924 (0.009)	0.362 (0.016)	0.365 (0.016)	0	0	0.14604855	GES (-)
2048	0.96 (0.007)	0.465 (0.017)	0.437 (0.017)	0	0	0	GES (MMHC)
8192	0.974 (0.005)	0.494 (0.017)	0.395 (0.016)	0	0	0	GES (MMHC)

Execution time							
Sample	GES	MMHC	LBNA	GES-MMHC	GES-LBNA	MMHC-LBNA	Significant winner
128	619.498 (421.802)	7.942 (1.446)	8.788 (3.019)	0	0	0	MMHC (MMHC)
512	369.641 (57.67)	9.156 (0.038)	10.124 (0.704)	0	0	0	MMHC (MMHC)
2048	636.54 (28.191)	15.545 (0.344)	22.786 (0.692)	0	0	0	MMHC (MMHC)
8192	1093.798 (75.179)	55.069 (5.868)	80.602 (6.786)	0	0	0	MMHC (MMHC)

Diabetes

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	633.156 (4.541)	633.742 (4.841)	0.3251543	-
512	84.758 (10.744)	31.945 (6.421)	0	LBNA
2048	52.57 (8.546)	6.586 (3.062)	0	LBNA
8192	497.445 (21.895)	309.641 (20.035)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.255 (0.011)	0.253 (0.012)	0.26416492	-
512	1.061 (0.008)	1.046 (0.008)	0	MMHC
2048	1.012 (0.005)	1.008 (0.003)	0	MMHC
8192	1.129 (0.013)	0.713 (0.016)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.727 (0.025)	0.722 (0.027)	0.08161831	-
512	0.942 (0.007)	0.955 (0.007)	0	LBNA
2048	0.988 (0.004)	0.992 (0.003)	0	LBNA
8192	0.616 (0.016)	0.903 (0.02)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.185 (0.009)	0.183 (0.01)	0.02645493	MMHC
512	0.999 (0.001)	0.999 (0.002)	0.56872749	-
2048	1 (0)	1 (0)	1	-
8192	0.695 (0.016)	0.644 (0.017)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1 (0)	1 (0)	1	-
512	0.994 (0.006)	0.999 (0.002)	0	LBNA
2048	0.992 (0.009)	0.999 (0.003)	0	LBNA
8192	0.376 (0.02)	0.882 (0.036)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0 (0)	0 (0)	1	-
512	0.917 (0.016)	0.992 (0.007)	0	LBNA
2048	0.921 (0.013)	0.997 (0.004)	0	LBNA
8192	0.474 (0.022)	0.5 (0.022)	0	LBNA

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.173 (0.004)	1.172 (0.004)	0.14493942	-
512	1.005 (0.001)	1 (0.001)	0	LBNA
2048	1.005 (0.001)	1 (0)	0	LBNA
8192	1.104 (0.005)	1.14 (0.006)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.5 (0.015)	0.498 (0.017)	0.33495522	-
512	0.664 (0.014)	0.664 (0.015)	0.83788586	-
2048	0.663 (0.015)	0.664 (0.016)	0.60042667	-
8192	0.619 (0.015)	0.601 (0.015)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	10.709 (5.165)	13.356 (0.885)	0	MMHC
512	15.444 (0.131)	37.644 (0.57)	0	MMHC
2048	18.718 (0.109)	43.103 (0.51)	0	MMHC
8192	74.568 (19.759)	113.15 (24.622)	0	MMHC

Pigs

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1324.328 (22.943)	1286.82 (18.831)	0	LBNA
512	1523.461 (39.882)	1303.773 (23.627)	0	LBNA
2048	1467.258 (39.38)	1080.625 (32.319)	0	LBNA
8192	1821.555 (28.17)	1249.711 (23.665)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.346 (0.023)	0.314 (0.017)	0	MMHC
512	0.695 (0.044)	0.465 (0.02)	0	MMHC
2048	1.083 (0.021)	0.751 (0.016)	0	MMHC
8192	1.157 (0.02)	0.599 (0.016)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.374 (0.024)	0.399 (0.026)	0	LBNA
512	0.374 (0.017)	0.486 (0.021)	0	LBNA
2048	0.461 (0.012)	0.631 (0.016)	0	LBNA
8192	0.297 (0.007)	0.486 (0.015)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.13 (0.012)	0.125 (0.01)	0.00173473	MMHC
512	0.259 (0.012)	0.226 (0.009)	0	MMHC
2048	0.5 (0.008)	0.473 (0.01)	0	MMHC
8192	0.344 (0.008)	0.291 (0.006)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.009 (0.01)	0.004 (0.009)	9.632e-05	MMHC
512	0.087 (0.016)	0.145 (0.034)	0	LBNA
2048	0.218 (0.019)	0.426 (0.046)	0	LBNA
8192	0.064 (0.007)	0.274 (0.034)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.001 (0.001)	0 (0.001)	1.91e-06	MMHC
512	0.026 (0.005)	0.016 (0.004)	0	MMHC
2048	0.162 (0.012)	0.18 (0.011)	0	LBNA
8192	0.066 (0.007)	0.075 (0.006)	0	LBNA

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.466 (0.005)	1.469 (0.004)	0	MMHC
512	1.568 (0.008)	1.641 (0.006)	0	MMHC
2048	1.335 (0.008)	1.401 (0.019)	0	MMHC
8192	1.609 (0.008)	1.769 (0.007)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.717 (0.015)	0.714 (0.015)	0.17067051	-
512	0.783 (0.013)	0.763 (0.013)	0	MMHC
2048	0.835 (0.011)	0.828 (0.012)	9.5e-07	MMHC
8192	0.79 (0.015)	0.76 (0.014)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	35.189 (1.364)	52.36 (1.463)	0	MMHC
512	44.051 (1.327)	74.722 (3.469)	0	MMHC
2048	80.11 (5.301)	172.188 (47.284)	0	MMHC
8192	152.351 (9.403)	294.929 (17.539)	0	MMHC

Link

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1370.289 (10.29)	1369.125 (10.956)	0.38569927	-
512	1608.258 (26.245)	1474.531 (12.06)	0	LBNA
2048	1785 (25.006)	1375.656 (15.262)	0	LBNA
8192	1659.312 (26.274)	1249.477 (18.008)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.127 (0.01)	0.126 (0.01)	0.27551556	-
512	0.389 (0.025)	0.256 (0.009)	0	MMHC
2048	0.74 (0.014)	0.328 (0.011)	0	MMHC
8192	0.851 (0.013)	0.426 (0.011)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.183 (0.024)	0.182 (0.027)	0.78809261	-
512	0.181 (0.015)	0.231 (0.019)	0	LBNA
2048	0.315 (0.013)	0.566 (0.019)	0	LBNA
8192	0.46 (0.011)	0.776 (0.017)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.023 (0.004)	0.023 (0.004)	0.39660835	-
512	0.07 (0.006)	0.059 (0.005)	0	MMHC
2048	0.233 (0.009)	0.185 (0.007)	0	MMHC
8192	0.391 (0.007)	0.331 (0.009)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.508 (0.502)	0.727 (0.447)	0.00042915	LBNA
512	0.037 (0.017)	0.175 (0.14)	0	LBNA
2048	0.026 (0.009)	0.227 (0.074)	0	LBNA
8192	0.044 (0.01)	0.329 (0.057)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0 (0)	0 (0)	0.62299633	-
512	0.005 (0.003)	0.003 (0.002)	0	MMHC
2048	0.018 (0.006)	0.017 (0.005)	0.16375446	-
8192	0.041 (0.009)	0.052 (0.009)	0	LBNA

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.475 (0.017)	1.474 (0.016)	0.46827793	-
512	1.39 (0.015)	1.427 (0.017)	0	MMHC
2048	1.266 (0.008)	1.388 (0.011)	0	MMHC
8192	1.177 (0.006)	1.314 (0.012)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.923 (0.009)	0.923 (0.008)	0.72631931	-
512	0.926 (0.008)	0.926 (0.008)	0.93268394	-
2048	0.934 (0.007)	0.931 (0.008)	0.00096798	MMHC
8192	0.938 (0.008)	0.936 (0.007)	0.03551102	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	77.412 (1.421)	95.242 (5.404)	0	MMHC
512	94.78 (2.332)	130.449 (2.855)	0	MMHC
2048	135.608 (4.79)	195.988 (5.677)	0	MMHC
8192	212.872 (8.065)	320.195 (8.915)	0	MMHC

Munin2

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1526.383 (10.407)	1523.25 (9.175)	0.01130009	LBNA
512	1784.906 (21.997)	1643.742 (12.238)	0	LBNA
2048	1959.469 (25.863)	1533.602 (11.354)	0	LBNA
8192	1982.164 (28.736)	1437.93 (15.966)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.121 (0.008)	0.118 (0.007)	0.00284576	MMHC
512	0.354 (0.017)	0.231 (0.009)	0	MMHC
2048	0.69 (0.015)	0.261 (0.011)	0	MMHC
8192	0.914 (0.014)	0.331 (0.011)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.165 (0.023)	0.164 (0.023)	0.55838203	-
512	0.152 (0.013)	0.188 (0.019)	0	LBNA
2048	0.286 (0.013)	0.53 (0.023)	0	LBNA
8192	0.347 (0.01)	0.693 (0.019)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.02 (0.003)	0.019 (0.003)	0.07537174	-
512	0.054 (0.005)	0.043 (0.005)	0	MMHC
2048	0.197 (0.008)	0.138 (0.008)	0	MMHC
8192	0.317 (0.008)	0.229 (0.009)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.32 (0.468)	0.578 (0.496)	5.341e-05	LBNA
512	0.008 (0.008)	0.068 (0.111)	0	LBNA
2048	0.03 (0.008)	0.328 (0.079)	0	LBNA
8192	0.052 (0.008)	0.324 (0.06)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0 (0)	0 (0)	1	-
512	0.001 (0.001)	0.001 (0.001)	4.864e-05	MMHC
2048	0.018 (0.004)	0.011 (0.003)	0	MMHC
8192	0.059 (0.008)	0.034 (0.007)	0	MMHC

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.558 (0.027)	1.558 (0.031)	0.92395592	-
512	1.541 (0.023)	1.591 (0.03)	0	MMHC
2048	1.355 (0.013)	1.59 (0.022)	0	MMHC
8192	1.264 (0.01)	1.622 (0.018)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.904 (0.009)	0.904 (0.009)	0.88277626	-
512	0.911 (0.008)	0.909 (0.009)	0.14957523	-
2048	0.923 (0.008)	0.918 (0.008)	1.24e-05	MMHC
8192	0.928 (0.009)	0.917 (0.009)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	89.18 (1.901)	99.167 (6.14)	0	MMHC
512	110.145 (2.4)	139.093 (2.908)	0	MMHC
2048	180.467 (9.215)	213.693 (5.199)	0	MMHC
8192	463.158 (43.616)	510.062 (27.351)	0	MMHC

Munin4

Structural Hamming Distance				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1442.602 (15.853)	1439.898 (15.477)	0.16952896	-
512	1747.336 (30.484)	1526.703 (12.785)	0	LBNA
2048	1957.594 (31.348)	1441.398 (12.198)	0	LBNA
8192	1848.539 (27.443)	1347.82 (19.046)	0	LBNA

Normalized number of edges				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.13 (0.012)	0.126 (0.011)	0.00661087	MMHC
512	0.438 (0.025)	0.239 (0.011)	0	MMHC
2048	0.818 (0.016)	0.311 (0.012)	0	MMHC
8192	0.964 (0.012)	0.403 (0.012)	0	MMHC

Skeleton precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.178 (0.033)	0.172 (0.034)	0.17367458	-
512	0.169 (0.014)	0.255 (0.024)	0	LBNA
2048	0.285 (0.013)	0.566 (0.022)	0	LBNA
8192	0.403 (0.01)	0.74 (0.02)	0	LBNA

Skeleton recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.023 (0.004)	0.022 (0.004)	0.00168037	MMHC
512	0.074 (0.006)	0.061 (0.006)	0	MMHC
2048	0.233 (0.009)	0.176 (0.009)	0	MMHC
8192	0.388 (0.008)	0.298 (0.008)	0	MMHC

V-structures precision				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.41 (0.492)	0.488 (0.5)	0.23139	-
512	0.016 (0.011)	0.147 (0.143)	0	LBNA
2048	0.017 (0.007)	0.149 (0.062)	0	LBNA
8192	0.04 (0.008)	0.213 (0.046)	0	LBNA

V-structures recall				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0 (0)	0 (0)	1	-
512	0.005 (0.003)	0.002 (0.002)	0	MMHC
2048	0.016 (0.006)	0.011 (0.004)	0	MMHC
8192	0.049 (0.009)	0.033 (0.007)	0	MMHC

Normalized score				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	1.54 (0.037)	1.546 (0.032)	0.20307636	-
512	1.456 (0.027)	1.533 (0.038)	0	MMHC
2048	1.288 (0.012)	1.506 (0.023)	0	MMHC
8192	1.166 (0.007)	1.447 (0.018)	0	MMHC

Prediction accuracy				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	0.921 (0.009)	0.919 (0.009)	0.08318329	-
512	0.925 (0.009)	0.925 (0.008)	0.78839302	-
2048	0.934 (0.008)	0.932 (0.009)	0.15362072	-
8192	0.939 (0.007)	0.933 (0.008)	0	MMHC

Execution time				
Sample	MMHC	LBNA	MMHC-LBNA	Significant winner
128	87.656 (1.764)	98.664 (6.501)	0	MMHC
512	113.791 (3.709)	145.56 (3.678)	0	MMHC
2048	184.48 (11.805)	252.638 (11.293)	0	MMHC
8192	456.531 (31.892)	557.246 (30.329)	0	MMHC

Munin3

Appendix B

Notation

We list here most of the notations used in this dissertation. Each next row of the following tables corresponds to one notation—namely it is placed in the first column.

There might potentially appear in general some additional indexes or suffixes in the used signs, like for example X_1 or $x1$ instead of respectively X and x. There can be used different signs or chains of signs in these elements, however they must follow the convention given in the table. For example instead of X and x there might appear respectively in this work sign Y or z. Moreover, sometimes there can be used in the place of some variables more complicated expressions corresponding exactly to them. For example instead of the general form $MC^{\mathcal{P}}_{\mathbb{X}}$ there appear in the work such variations like $MC^{\mathcal{P}}_{\{X\}\cup MB^{\mathcal{P}}_X}$.

Two occurring here notations, $\mathcal{P}$ and $\mathcal{V}$ are the special ones in the sense that they are in almost whole this work fixed (the exception occurs in Section 3.3 where we additionally parametrize them—however we still use letters $\mathcal{P}$ and $\mathcal{V}$, but with the appropriate indexes). These two signs are reserved for corresponding to them notions.

The second column is the short description of the concept, and the third column is the reference to some concrete place in the dissertation, where the first occurrence can be found, usually corresponding to some further more detailed description.

Note that many introduced in this dissertation concepts do not occur in the following tables. The convention applied here is that we list only these concepts which appear in several distant places of work—where the recognition of some used much earlier symbol might be indeed problematic.

All concepts are ordered according to the last column, that is according to their appearance.

Notation	Description	Reference
$\mathcal{V}$	The set of all considered random variables / the set of all attributes in the corresponding dataset / the set of all nodes of the corresponding Bayesian network structure.	Subsection 2.1.1
$\mathcal{D} = (\mathcal{U}, \mathcal{V})$	The information system / dataset $\mathcal{D}$ consisting of the set of objects $\mathcal{U}$ and the set of attributes $\mathcal{V}$.	Definition 2.1.1.1
X	The element of $\mathcal{V}$.	Subsection 2.1.1
$\mathbb{X}$	The subset of $\mathcal{V}$.	Subsection 2.1.1
$\mathcal{P}$	The joint probability distribution of the set of variables $\mathcal{V}$.	Subsection 2.1.2
$\mathcal{P}_{\mathcal{D}}$	The joint probability distribution inherited from the frequencies occurring in the given dataset $\mathcal{D}$.	Subsection 2.1.2
Dom_X	The set of possible values of the random variable/dataset attribute/graph node $X \in \mathcal{V}$.	Subsection 2.2.1
x	The element of Dom_X for some $X \in \mathcal{V}$.	Subsection 2.2.1
$Dom_{\mathbb{X}}$	The set of possible values configurations of the random variables/dataset attributes/graph nodes subset $\mathbb{X} \subseteq \mathcal{V}$.	Subsection 2.2.1
$\mathbb{x}$	The element of $Dom_{\mathbb{X}}$ for some $\mathbb{X} \in \mathcal{V}$.	Subsection 2.2.1
$Ind_{\mathcal{P}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$	The indicator that for some disjoint subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subset \mathcal{V}$ where $\mathbb{X} \neq \emptyset$ and $\mathbb{Y} \neq \emptyset$ there holds the conditional independence of $\mathbb{X}$ and $\mathbb{Y}$ given $\mathbb{Z}$ according to the joint probability distribution $\mathcal{P}$ of $\mathcal{V}$.	Definition 2.2.1.1
$Ind_{\mathcal{P}}$	The set of all conditional independencies existing in the given joint probability distribution $\mathcal{P}$ of $\mathcal{V}$.	Definition 2.2.1.1
χ^2	The χ^2 test for conditional independence.	Subsection 2.2.2
G^2	The G^2 test for conditional independence.	Subsection 2.2.2
$Assoc_{\mathcal{P}}(X, Y \mid \mathbb{Z})$	The measure of conditional association basing on a conditional independence test used in several presented in this work constraint-based algorithms.	Equation 2.2.6.1
df^S_{adj}	The Spirtes et al. (2000) adjusted number of degrees of freedom used in the G^2 test in the case of structural zeros in the considered conditional relation.	Subsection 2.2.7
df^T_{adj}	The Tsamardinos et al. (2006) adjusted number of degrees of freedom used in the G^2 test in the case of structural zeros in the considered conditional relation.	Subsection 2.2.7

Notation	Description	Reference
$MB_T^{\mathcal{P}}$	The Markov blanket of some variable $T \in \mathcal{V}$ with regard to the joint probability distribution $\mathcal{P}$ of $\mathcal{V}$.	Definition 2.2.9.1
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The graph $\mathcal{G}$ consisting of the set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}$.	Subsection 2.3.1
$\mathcal{BN} = (\mathcal{G}, \mathcal{P})$	The Bayesian network $\mathcal{BN}$ based on the structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and representing the joint probability distribution $\mathcal{P}$ of $\mathcal{V}$.	Definition 2.3.1.1
$\pi_X^{\mathcal{G}}$	The set of parents of the vertex X in the graph $\mathcal{G}$.	Definition 2.3.1.2
$[X_1, ..., X_k]_{\mathcal{G}}$	The chain between vertices X_1 and X_k in the graph $\mathcal{G}$.	Definition 2.3.4.2
$Ind_{\mathcal{G}}(\mathbb{X}, \mathbb{Y} \mid \mathbb{Z})$	The indicator that for some disjoint subsets $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \subset \mathcal{V}$ where $\mathbb{X} \neq \emptyset$ and $\mathbb{Y} \neq \emptyset$ there holds the d-separation of $\mathbb{X}$ and $\mathbb{Y}$ given $\mathbb{Z}$ in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.	Definition 2.3.4.3
$Ind_{\mathcal{G}}$	The set of all d-separations occurring in the given graph $\mathcal{G}$.	Definition 2.3.4.3
$Score(\mathcal{D}, \mathcal{G})$	The scoring criterion $Score$ evaluating the Bayesian network structure $\mathcal{G}$ in the context of the dataset $\mathcal{D}$.	Subsection 3.2.4
ST_1, ST_2, SP_1, SP_2	Two theoretical and two practical scenarios of Bayesian network structure learning, on the basis of which the performance analysis of several algorithms is given in this dissertation.	Subsection 4.1.3
SGS	The SGS Bayesian network structure learning algorithm.	Algorithm 4.2.3.1
PC	The PC Bayesian network structure learning algorithm.	Algorithm 4.2.4.1
b_m	The commonly used value depending on the number of objects m of the input dataset equal to $b_m = \lfloor \log_2 m - (\log_2 c + 2) \rfloor$ where c is the user-defined constant. It expresses the largest possible conditional set for which the statistical test for conditional independence is normally conducted in the practical scenario SP_2.	Subsection 4.2.8
GS	The GS Markov blanket learning algorithm and Bayesian network structure learning mechanism basing on learned Markov blankets.	Algorithm 4.3.3.1, 4.3.4.1
IAMB	The IAMB Markov blanket learning algorithm.	Algorithm 5.1.1.1

Notation	Description	Reference
GS-GS	The Bayesian network structure learning algorithm resulting from applying the GS Markov blanket learning method as a subroutine in the GS Bayesian network structure construction from Markov blankets mechanism.	Subsection 5.1.4
GS-IAMB	The Bayesian network structure learning algorithm resulting from applying the IAMB Markov blanket learning method as a subroutine in the GS Bayesian network structure construction from Markov blankets mechanism.	Subsection 5.1.4
MMPC	The neighbors (parents and children) learning algorithm.	Algorithm 5.2.2.1
MMHC	The MMHC Bayesian network structure learning algorithm.	Algorithm 5.3.1.1, 5.3.5.1
$\mathcal{P}\mid_{\mathbb{X}}$	The joint probability distribution of the set of variables $\mathbb{X}$ implied by the distribution $\mathcal{P}$.	Subsection 6.2.1
$V(\mathcal{C})$	The set of vertices in the structure characterizing the Markov equivalence class $\mathcal{C}$.	Subsection 6.2.1
$E(\mathcal{C})$	The set of edges in the skeleton characterizing the Markov equivalence class $\mathcal{C}$.	Subsection 6.2.1
$MC_{\mathbb{X}}^{\mathcal{P}}$	The Markov equivalence class of all perfect maps of $\mathcal{P}\mid_{\mathbb{X}}$ for the subset $\mathbb{X} \subseteq \mathcal{V}$ and joint probability distribution $\mathcal{P}$ of $\mathcal{V}$.	Subsection 6.2.1
LBNA	The theoretical version of the LBNA algorithm.	Algorithm 6.3.1.1
LBNA($\mathcal{MC}$, $\mathcal{MB}$)	The practical version of the LBNA algorithm, with the auxiliary Bayesian network structure learning method $\mathcal{MC}$ and Markov blanket learning approach $\mathcal{MB}$. Note that in Chapter 7 and Appendix A we call the LBNA(MMHC, MMHC) method shortly as LBNA.	Algorithm 6.4.3.1, 6.4.4.1
GES	The GES Bayesian network structure learning algorithm.	Subsection 7.1.1
$Sk(\mathcal{BN})$	The set of edges occurring in the skeleton of the Bayesian network $\mathcal{BN}$.	Subsection 7.1.5
$Vs(\mathcal{BN})$	The set of v-structures occurring in the structure of the Bayesian network $\mathcal{BN}$.	Subsection 7.1.5

References

B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–71, 1996.

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON: A Novel Markov Blanket Algorithm for Optimal Variable Selection. *AMIA Annual Symposium Proceedings*, 2003:21–25, 2003.

C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos. Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification. Part I: Algorithms and Empirical Evaluation. *Journal of Machine Learning Research*, 11: 171–234, 2010a.

C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Koutsoukos. Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification. Part II: Analysis and Extensions. *Journal of Machine Learning Research*, 11:235–284, 2010b.

S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen. MUNIN - an Expert EMG Assistant. In J. E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, pages 255–277. Elsevier Science, Amsterdam, NL, 1989.

S. Andreassen, R. Hovorka, J. Benn, K. G. Olesen, and E. R. Carson. A Model-Based Approach to Insulin Adjustment. In *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, pages 239–248, Maastricht, NL, 1991. Springer-Verlag.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 247–256, Berlin, Heidelberg, DE, 1989. Springer-Verlag.

P. Betliński. Markov Blanket Approximation Based on Clustering. In *Foundations of Intelligent Systems. 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings*, pages 192–202, Berlin, Heidelberg, DE, 2011. Springer-Verlag.

P. Betliński and D. Ślęzak. The Problem of Finding the Sparsest Bayesian Network for an Input Data Set is NP-Hard. In *Foundations of Intelligent Systems. 20th International Symposium, ISMIS 2012, Macau, China, December 4-7, 2012. Proceedings*, pages 21–30, Berlin, Heidelberg, DE, 2012. Springer-Verlag.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2–3):213–244, 1997.

J. Cheng and M. J. Druzdzel. AIS-BN: An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

D. M. Chickering. Learning Bayesian Networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130, New York, NY, 1996. Springer-Verlag.

D. M. Chickering. Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3:507–554, 2002.

D. M. Chickering, D. Heckerman, and C. Meek. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel. On-Line Student Modeling for Coached Problem Solving Using Bayesian Networks. In *Proceedings of the Sixth International Conference on User Modeling*, pages 231–242, Sardinia, IT, 1997. Springer-Verlag.

G. F. Cooper. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

G. F. Cooper, C. F. Aliferis, R. Ambrosino, J. Aronis, B. G. Buchanan, R. Caruana, M. J. Fine, C. Glymour, G. Gordon, and B. H. Hanusa. An Evaluation of Machine-Learning Methods for Predicting Pneumonia Mortality. *Artificial Intelligence in Medicine*, 9(2): 107–138, 1997.

M. J. Druzdzel and M. Henrion. Efficient Reasoning in Qualitative Probabilistic Networks. In *Proceedings of the Eleventh Annual Conference on Artificial Intelligence (AAAI)*, pages 548–553, Washington, DC, 1993.

M. J. Druzdzel and L. C. van der Gaag. Elicitation of Probabilities for Belief Networks: Combining Qualitative and Quantitative Information. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 141–148, Montreal, QU, 1995. Morgan Kaufmann.

R. A. Fisher. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 206–215, San Francisco, CA, 1999. Morgan Kaufmann.

R. M. Fung and K.-C. Chang. Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 209–220, Amsterdam, NL, 1989. Elsevier Science.

P. I. Good. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*, volume 2. Springer, New York, NY, 2000.

D. E. Heckerman, E. J. Horvitz, and B. N. Nathwani. Towards Normative Expert Systems: Part I. The Pathfinder Project. *Methods of Information in Medicine*, 31(2):90–105, 1992.

D. E. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

M. Henrion. Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. In *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 149–164, Amsterdam, NL, 1986. Elsevier Science.

M. Henrion and M. J. Druzdzel. Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 10–20, Amsterdam, NL, 1990. Elsevier Science.

K. Höffgen. Learning and Robust Learning of Product Distributions. Technical Report 464, Fachbereich Informatik, Universität Dortmund, Dortmund, DE, 1993.

A.L. Jensen and F.V. Jensen. MIDAS—An Influence Diagram for Management of Mildew in Winter Wheat. In *Proceedings of the Twelfeth Conference on Uncertainty in Artificial Intelligence*, pages 349–356, San Francisco, CA, 1996. Morgan Kaufmann.

C. S. Jensen and A. Kong. Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops. Technical Report R-96-2048, Department of Computer Science, Aalborg University, Denmark, Aalborg, DK, 1996.

F. V. Jensen, U. Kjærulff, K. G. Olesen, and J. Pedersen. Et forprojekt til et ekspertsystem for drift af spildevandsrensning (An Expert System for Control of Waste Water Treatment—A Pilot Project. Technical report, Judex Datasystemer A/S, Aalborg, DK, 1989. In Danish.

J. H. Kim and J. Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 190–193, Karlsruhe, DE, 1983.

D. Koller and M. Sahami. Towards Optimal Feature Selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, Bari, IT, 1996.

K. Kristensen and I. A. Rasmussen. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33(3):197–217, 2002.

S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 50(2):157–224, 1988.

D. Margaritis. *Learning Bayesian Network Model Structure from Data.* PhD thesis, Carnegie Mellon University, 2003.

D. Margaritis and S. Thrun. Bayesian Network Induction via Local Neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511, Cambridge, MA, 2000. MIT Press.

C. Meek. Strong completeness and faithfulness in Bayesian networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 411–418, San Francisco, CA, 1995. Morgan Kaufmann.

C. Meek. *Graphical Models: Selecting causal and statistical models.* PhD thesis, Carnegie Mellon University, 1997.

R. Nagarajan, M. Scutari, and S. Lebre. *Bayesian Networks in R with Applications in Systems Biology.* Springer, New York, 2013.

R. E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, Upper Saddle River, NJ, 2003.

A. Oniśko. *Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders.* PhD thesis, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, 2003.

C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

Z. Pawlak. *Information Systems: Theoretical Foundations.* WNT, 1983.

Z. Pawlak and A. Skowron. Rudiments of Rough Sets. *Information Sciences*, 177(1):3–27, 2007.

J. Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 133–136, Menlo Park, CA, 1982. AAAI Press.

J. Pearl. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. In *Proceedings of the Seventh Conference of the Cognitive Science Society*, pages 329–334, Irvine, CA, 1985. University of California.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

J. Pearl, D. Geiger, and T. S. Verma. The Logic of Influence Diagrams. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 67–88, Toronto, CDN, 1990. Wiley.

K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine Series 5*, 50(302):157–175, 1900.

J.-P. Pellet and A. Elisseeff. Using Markov Blankets for Causal Structure Learning. *Journal of Machine Learning Research*, 9:1295–1342, 2008.

R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2014. URL http://www.R-project.org/.

M. Scutari. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3):1–22, 2010.

M. Scutari. *Measures of Variability for Graphical Models.* PhD thesis, Ph.D. School in Statistical Sciences, University of Padova, 2011.

R. D. Shachter and M. A. Peot. Simulation Approaches to General Probabilistic Inference on Belief Networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 221–231, Amsterdam, NL, 1989. Elsevier Science.

D. Ślęzak. Approximate Entropy Reducts. *Fundamenta Informaticae*, 53(3–4):365–390, 2002.

D. Ślęzak. Degrees of conditional (in)dependence: A framework for approximate Bayesian networks and examples related to the rough set-based feature selection. *Information Sciences*, 179(3):197–209, 2009.

D. J. Spiegelhalter and R. G. Cowell. Learning in probabilistic expert systems. In J. M. Bernardo, J. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 4*, pages 447–465, Oxford, UK, 1992. Oxford University Press.

P. Spirtes and C. Glymour. An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. Causality from Probability. In J. E. Tiles, G. T. McKee, and G. C. Dean, editors, *Evolving Knowledge in Natural Science and Artificial Intelligence*, pages 181–199, London, GB, 1990. Pitman.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search*. The MIT Press, Cambridge, MA, 2000.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Algorithms for Large Scale Markov Blanket Discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 376–380, Palo Alto, CA, 2003a. AAAI Press.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 673–678, New York, NY, 2003b. ACM.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

M. P. Wellman. Fundamental Concepts of Qualitative Probabilistic Networks. *Artificial Intelligence*, 44(3):257–303, 1990.

R. Yehezkel and B. Lerner. Recursive autonomy identification for Bayesian network structure learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 429–436, Bridgetown, BB, 2005.

R. Yehezkel and B. Lerner. Bayesian Network Structure Learning by Recursive Autonomy Identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.